WO2003088095

Publication Title:

No title available

Abstract:

Abstract not available for WO2003088095

Data supplied from the esp@cenet database - Worldwide

Courtesy of http://v3.espacenet.com

(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2003 年10 月23 日 (23.10.2003)

PCT

(10) 国際公開番号
WO 03/088095 A1

(51) 国際特許分類[7]: G06F 17/50, H03K 19/173, H01L 21/82

(21) 国際出願番号: PCT/JP03/04787

(22) 国際出願日: 2003 年4 月15 日 (15.04.2003)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願2002-115273 2002 年4 月17 日 (17.04.2002) JP
特願2002-147930 2002 年5 月22 日 (22.05.2002) JP

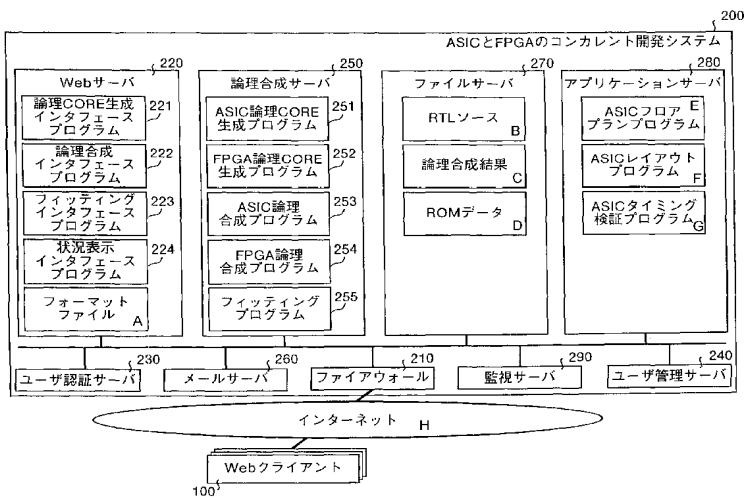
(71) 出願人 (米国を除く全ての指定国について): 富士通株式会社 (FUJITSU LIMITED) [JP/JP]; 〒211-8588 神奈川県 川崎市中原区 上小田中４丁目１番１号 Kanagawa (JP).

(72) 発明者; および
(75) 発明者/出願人 (米国についてのみ): 古賀 智昭 (KOGA,Chiaki) [JP/JP]; 〒812-0011 福岡県 福岡市博多区 博多駅前三丁目２２番８号 富士通九州ディジタル・テクノロジ株式会社内 Fukuoka (JP). 津田 昌行 (TSUDA,Masayuki) [JP/JP]; 〒812-0011 福岡県 福岡市博多区 博多駅前三丁目２２番８号 富士通九州ディジタル・テクノロジ株式会社内 Fukuoka (JP). 中山 彰二 (NAKAYAMA,Akitsugu) [JP/JP]; 〒812-0011 福岡県 福岡市博多区 博多駅前三丁目２２番８号 富士通九州ディジタル・テクノロジ株式会社内 Fukuoka (JP).

[続葉有]

(54) Title: INTEGRATED CIRCUIT DEVELOPMENT METHOD, PROGRAM STORAGE MEDIUM CONTAINING INTEGRATED CIRCUIT DEVELOPMENT METHOD, SYSTEM FOR CONCURRENT DEVELOPMENT OF ASIC AND PROGRAMMABLE LOGIC DEVICE, DEVELOPMENT PROGRAM, AND DEVELOPMENT METHOD

(54) 発明の名称: 集積回路の開発方法及び集積回路の開発方法を記憶したプログラム記憶媒体、ならびにＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム、開発プログラム及び開発方法



200...ASIC AND FPGA CONCURRENT DEVELOPMENT SYSTEM
220...Web SERVER
221...LOGICAL CORE GENERATION INTERFACE PROGRAM
222...LOGICAL SYNTHESIS INTERFACE PROGRAM
223...FITTING INTERFACE PROGRAM
224...STATE DISPLAY INTERFACE PROGRAM
A...FORMAT FILE
250...LOGICAL SYNTHESIS SERVER
251...ASIC LOGICAL CORE GENERATION PROGRAM
252...FPGA LOGICAL CORE GENERATION PROGRAM
253...ASIC LOGICAL SYNTHESIS PROGRAM
254...FPGA LOGICAL SYNTHESIS PROGRAM
255...FITTING PROGRAM
270...FILE SERVER
B...RTL SOURCE
C...LOGICAL SYNTHESIS RESULT
D...ROM DATA
280...APPLICATION SERVER
E...ASIC FLOOR PLAN PROGRAM
F...ASIC LAYOUT PROGRAM
G...ASIC TIMING VERIFICATION PROGRAM
230...USER AUTHENTICATION SERVER
260...MAIL SERVER
210...FIRE WALL
290...MONITORING SERVER
240...USER MANAGEMENT SERVER
H...INTERNET
100...Web CLIENT

(57) Abstract: An integrated circuit development method for generating a net list called a core (logical core) composed of a net connecting ports of a block not depending on a device technology by using only connection information in the block port specification which is a result of circuit architecture study and a part of logical design document, selecting blocks as objects from the core (logical core), grouping the blocks, and using the grouped core (logical core) data. A system for concurrent development of ASIC and FPGA is constituted by a fire wall for monitoring access from the Internet, a Web server for communicating with the Web client used by a user, an authentication server for performing user authentication, a user management server for managing a user, a logical synthesis server for executing an ASIC and FPGA development program, a mail server for distributing a mail to those associated with the project, a file server for storing

[続葉有]

WO 03/088095 A1

(74) 代理人: 酒井 宏明 (SAKAI,Hiroaki); 〒100-0013 東京都 千代田区 霞が関三丁目２番６号 東京倶楽部ビルディング Tokyo (JP).

(81) 指定国 (国内): CN, JP, KR, US.

添付公開書類:
— 国際調査報告書

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

design information, an application server for executing an ASIC implement design program, and a monitoring server for monitoring the ASCI and FPGA development state.

(57) 要約: 回路アーキテクチャ検討結果であり論理設計ドキュメントの一部であるブロックのポート仕様の接続情報のみを用いてデバイス・テクノロジに依存しないブロックのポートとポート間を結ぶネットからなるコア（論理コア）と呼ぶネットリストを生成し、コア（論理コア）から対象ブロックを選択、グループ化し、グループ化したコア（論理コア）のデータを使用する集積回路の開発方法、およびインターネットからのアクセスを監視するファイアウォールと、ユーザが使用するＷｅｂクライアントと通信するＷｅｂサーバと、ユーザの認証を行なう認証サーバと、ユーザを管理するユーザ管理サーバと、ＡＳＩＣとＦＰＧＡの開発用プログラムを実行する論理合成サーバと、Ｐｒｏｊｅｃｔの関係者にメールを配信するメールサーバと、設計情報を格納するファイルサーバと、ＡＳＩＣのインプリメント設計用プログラムを実行するアプリケーションサーバと、ＡＳＩＣとＦＰＧＡの開発状況を監視する監視サーバとから構成されるＡＳＩＣとＦＰＧＡのコンカレント開発システム。

明　細　書

集積回路の開発方法及び集積回路の開発方法を記憶したプログラム記憶媒体、ならびにＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム、開発プログラム及び開発方法

技術分野

本発明は集積回路の開発方法、集積回路の開発方法を記憶したプログラム記憶媒体及び論理合成ツールの制御装置、ならびにネットワークに接続されたコンピュータからユーザが利用するＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム、開発プログラム及び開発方法に関する。

更に詳しくは、集積回路の開発におけるプログラマブル・ロジックデバイス構成方法に係り、回路アーキテクチャ検討結果であり論理設計ドキュメントの一部であるブロックのポート仕様の接続情報のみを用いてデバイス・テクノロジに依存しないブロックのポートとポート間を結ぶネットからなるコア（論理コア）と呼ぶネットリストを生成し、コア（論理コア）から対象ブロックを選択、グループ化し、グループ化したコア（論理コア）のデータを使用し、集積回路の開発を行なう集積回路の開発方法、ならびにＡＳＩＣとプログラマブル・ロジックデバイスのシームレスなコンカレント開発を可能とし、設計品質確保と開発期間短縮を両立するとともに、開発にかかる人的資源とコストを低減することができるＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム、開発プログラム及び開発方法に関する。

背景技術

集積回路の手順では、先ず仕様を決定する際、もれ等がないか慎重に検討した後、その仕様に従い設計を行なうようになっている。第１１図は集積回路設計の手順を示すフローチャートである。先ず、製品仕様からＡＳＩＣ（特定用途向け

ＩＣ）のスペックを取得する（ｇ１）。次に、回路アーキテクチャを検討する（ｇ２）。次に、回路アーキテクチャ検討結果に基づいて回路設計を行なう（ｇ３）。この回路設計は、論理検証（ｇ４）を行ないながら行なう。

次に、回路設計が終了したら回路の論理合成を行なう（ｇ５）。論理合成が終了したら、論理合成結果に基づいてレイアウト展開する（ｇ６）。この段階で回路が設計できたことになる。回路が設計できたら、当該回路を製造し（ｇ７）、製造した回路を用いて実機の評価を行なう（ｇ８）。以上のシーケンスの内、本発明に係る部分は、ステップｇ５の論理合成に係る部分である。

集積回路開発において、仕様を入力として実現する機能の検討、機能を実現する回路構成の検討を行なう回路アーキテクチャ検討を以下のような流れで実施している。通常、集積回路の開発では、製品の仕様から製品を実現する機能を洩れなく洗い出し、洗い出した機能を実現する回路の構成の検討や、ＩＰ（Ｉｎｔｅｌｌｅｃｔｕａｌ　Ｐｒｏｐｅｒｔｙ：知的財産）等のマクロの検討を行なう。ここで、マクロとはＩＰも含むＲＡＭ、ＲＯＭ等の変更しなくても使えるものをいう。

構成を検討した回路やＩＰの実現規模の初期見積りを行ない、この時予め判っていればゲート数で算出し、ゲート数が判らない場合は必要な信号数と処理に必要な時間からフリップフロップ数を算出して実現規模を見積もっている。ここで、見積もった規模と各機能の入出力信号（以降ポートと称する）本数を基に複数の機能をグループ化し１つのブロックとする。このグループ化を全機能について行なう。

論理設計では、前述の機能と見積り規模を基にＨＤＬ（ハードウェア記述言語）等の手段によりプログラマブル・ロジックデバイスを対象に回路設計を行ない、オンボードでの機能評価が行なわれる。評価完了後、ＡＳＩＣ化する場合に再設計及び再検証が行なわれていた。

機能評価の完了後、ＡＳＩＣ化によるコストダウンが実施される場合、プログラマブル・ロジックデバイス（例えばＦＰＧＡ）を対象に設計する時点からＡＳ

ＩＣを考慮した設計を施されることが少なく、Ｉ／Ｏバッファ、デバイス用試験回路、メモリ等のマクロなど、プログラマブル・ロジックデバイスとＡＳＩＣ間の違いのためＡＳＩＣを対象にプログラマブル・ロジックデバイスの設計データを基に再設計が発生し、再設計することによる設計データの二重管理、再設計と機能の再検証による開発期間の長期化と開発コストの増大が表面化してきている。

ここで、ＡＳＩＣは開発期間が長いがコストが安いという特徴があり、一方、プログラマブル・ロジックデバイス（ＦＰＧＡ）は開発期間は短いがコストが高いという特徴がある。

本発明はこのような課題に鑑みてなされたものであって、大規模ＡＳＩＣの開発における論理設計と、論理合成、レイアウトのコンカレント（並行した）開発に適用している回路アーキテクチャ検討において、チップを機能的に分割したブロックのポート情報と、チップのポート情報とからポート間の接続情報としてブロック間ネットリストを生成する方法（特開２０００－９０１４２号に記載）を応用した、集積回路の開発方法及び該集積回路の開発方法によって開発する集積回路を構成するブロックとブロック間ネットをブロック間ネットリストから任意の規模と個数生成するよう論理合成ツールを制御する装置を提供し、アーキテクチャの共有化を実現し、再設計と再検証を極力回避することができる集積回路の開発方法及び集積回路の開発方法を記憶したプログラム記憶媒体を提供することを目的としている。

また、近年、半導体の微細化により１０Ｍゲートを超すＡＳＩＣも開発可能となっているが、電子機器の高機能化と複雑化に伴い、仕様設計，論理設計およびフロアプラン、論理合成，レイアウト設計、タイミング検証を行なうインプリメント設計が長期化すると共に設計品質の確保が困難となってきている。特に、ＡＳＩＣ開発のリメイクは、電子機器の開発期間の長期化のみならずコストの増大、市場投入の機会損失を招いている。

このため、開発ＴＡＴ（Turn Around Time）が短く、設計変更が容易なプログ

ラマブル・ロジックデバイスが多用されつつあるが、プログラマブル・ロジックデバイスはコストがかかり、小型化が困難であることから、まずプログラマブル・ロジックデバイスで機能を実現し、プロトタイピングにてデバッグした後、量産時期にＡＳＩＣ化することが多い。

しかしながら、ＡＳＩＣ化を前提にプログラマブル・ロジックデバイスのプロトタイピングにより検証したとしても、プログラマブル・ロジックデバイスからＡＳＩＣへとシリアルな開発では、総開発手番を短くする事は難しいという問題がある。特に、ＡＳＩＣ開発時のインプリメント設計においてタイミング問題が発生すると、プログラマブル・ロジックデバイスの再設計からやり直しとなる可能性があり、半導体ベンダなど社外に依託している場合は、依託先の人的リソースの確保、人的リソースの確保に伴うコストの増大を招く。

更には、プログラマブル・ロジックデバイスとＡＳＩＣデバイス間の構造の違いからＡＳＩＣ専用に再設計が発生した場合は、プログラマブル・ロジックデバイスによるデバッグが無意味になるばかりでなく、開発期間が長期化してコストも増大する。どちらにしても市場投入の機会損失になる。

なお、ＡＳＩＣの大規模化に伴う開発期間の長期化の対策として、特開２０００－９０１４２号公報に記載のとおり、回路アーキテクチャ検討、論理設計・検証、インプリメント設計をコンカレントに行っているが、電子機器の機能の複雑化と市場の動きの早さによって、仕様設計、論理の設計、検証が長期化し開発手番を短くする事が困難となってきている。さらに、コンカレントに開発を進める場合には、ＡＳＩＣ開発の知識を有した人的資源と開発ツールが必要であり、半導体技術の進歩に伴い複雑化する開発ツールの教育も必要となるという問題が生ずる。

従って、この発明は、ＡＳＩＣとプログラマブル・ロジックデバイスのシームレスなコンカレント開発を可能とし、設計品質確保と開発期間短縮を両立するとともに、開発にかかる人的資源とコストを低減することができるＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム、開発プログラム及

び開発方法を提供することも目的とする。

発明の開示

第１図は本発明方法の原理を示すフローチャートである。請求の範囲第１項記載の発明は、ブロックのポートとポートの接続情報なるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックを選択しグループ化する手段により任意の規模、個数のブロックのポートとポートの接続情報からなる、論理合成ツールが読み込み可能なＨＤＬフォーマットのコア（論理コア）を生成し（ステップ１）、チップの端子情報から論理合成ツールで仮のチップデザインを作成し、このデザインに端子を発生させ（ステップ２）、作成したデザイン内部にステップ２と同一のデザインをセルとして発生させ（ステップ３）、デザインとセル間の同一名称のポートを接続し（ステップ４）、接続したポート間のネットに対し、デバイス・テクノロジに依存するＩ／Ｏバッファを挿入し（ステップ５）、ステップ１で作成したコア（論理コア）とセルを入れ替え、トップ階層であるデザインの階層を展開し、ネットリストを生成する（ステップ６）、ことを特徴とする。

請求の範囲第１項記載の発明では、ポート名、レンジ、入出力定義からブロックのポート仕様であるエンティティを作成しファイルに出力する（ファイルに書き込む）。入出力で出力定義されているポートに関して出力信号ファイルを作成する手段により、あるブロックの入力ポートに指定された、出力元インスタンスの出力ポート名に間違いがないかチェックし、出力元インスタンス名にデバイスのパッケージの端子と接続する旨のＩ／Ｏ等、キーワードの定義がある場合に、出力元インスタンスの出力ポート名にある名称をデバイスのパッケージの端子名とし、該端子が１本なのか複数（ベクタ）なのかをレンジの定義から判断する手段と、該端子が入力なのか出力なのかを入出力の定義から判断する手段と、該端子が双方向なのかを種別の定義から判断する手段により、コア（論理コア）のポート仕様であるエンティティを作成し、ファイルに出力する。ここで、インスタ

ンスとは回路ユニットのことである。

ブロックのポートが入出力定義で入力定義されている場合、該ポートの出力元インスタンス名と出力元インスタンスの出力ポート名の組が前述の出力信号ファイルに存在するかチェックする手段と、前記チェック結果、インスタンス間のポートの接続が可能と判断した場合、インスタンス間を接続する信号を生成し、ファイルに出力する。全インスタンスのチェックを含め、処理が完了した時、前述のコア（論理コア）のエンティティとインスタンス間ネットを読み込み、コア（論理コア）を生成しファイルに出力する。

チェック結果が問題なければ、ブロックの入出力ポート仕様とインスタンス間の接続ネットとインスタンスとデバイスパッケージの端子となる外部端子間の接続ネットからなり、論理設計部分を全く持たないＨＤＬ（ハードウェア記述言語）のファイル（以降コア（論理コア）と称する）を出力する手法をコンピュータに実行させるプログラムを記録したコンピュータが読み取り可能な記録媒体である。

従って、請求の範囲第１項の発明のコア（論理コア）を生成する手法をコンピュータに実行させるプログラムを記憶したプログラム記憶媒体によれば、集積回路開発において、ＲＴＬ設計の入力となるブロックのポート仕様の品質を事前に確保できる効果があり、更に機能ブロックが多く存在し、設計リソースも多い大規模集積回路の開発においても、事前にブロック間の接続が確認できていることにより、必ずチップが組み上がることを保証することができる。

請求の範囲第２項記載の発明は、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発を行なう場合において、ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックを選択しグループ化する手段により、任意の規模、個数のブロックのポートとポートの接続情報からなるコア（論理コア）を生成し（ステップ１）、チップの端子情報から論理合成ツールで仮のチップのデザインを作成し（ステップ２）、このデザインに前記チップの端子情報にある名称で端子を発生させ（ステップ３）

、この内部に前記と同一デザインをセルとして発生させ（ステップ４）、ステップ２で作成したデザインとセル間の同一名称のポートを接続し（ステップ５）、この接続ネットに対し、チップの端子情報からデバイス・テクノロジに依存するＩ／Ｏバッファを挿入し（ステップ６）、ステップ１で生成したコア（論理コア）を前記セルと入れ替え（ステップ７）、このコア（論理コア）のトップ階層である、作成したデザインの階層を展開しプログラマブル・ロジックデバイスのチップのネットリストを生成する（ステップ８）、ことを特徴とする。

請求の範囲第２項記載の発明は、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発において、請求の範囲第１項によるＡＳＩＣのコア（論理コア）から、論理合成ツールの機能を、後述する請求の範囲第３項及び請求項４記載のネットリスト生成手法をコンピュータに実行させるプログラムを記憶したプログラム記憶媒体を搭載した制御装置によって制御し、接続関係のある任意のブロックをグループ化し、プログラマブル・ロジックデバイスのコア（論理コア）として切り出し、プログラマブル・ロジックデバイスに依存するＩ／Ｏバッファを挿入し、プログラマブル・ロジックデバイスのチップのネットリストを生成することを特徴とする集積回路の開発方法である。

従って、請求の範囲第２項記載の集積回路の開発方法によれば、ＡＳＩＣのコア（論理コア）からプログラマブル・ロジックデバイスのコア（論理コア）としてブロック間接続を保持したまま切り出すため、本コア（論理コア）を用いたプログラマブル・ロジックデバイスの検証を実施すれば、ＡＳＩＣにおいては、少なくとも同一の構成での検証の重複を回避することができる。よって、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発を効率よく進めることができる。

また、Ｉ／Ｏバッファを挿入する際、チップの端子情報から仮のコア（論理コア）を生成し、ブロックから生成したコア（論理コア）と入れ替える手段によって、ブロックに定義されているチップのポート情報とチップの端子情報をクロスチェックでき、ブロックのポート仕様とチップ端子仕様の両者の品質を確保する

ことができる。

請求の範囲第３項記載の発明は、ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックを選択しグループ化する手段により、任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストを生成することを特徴とする。

このように構成すれば、任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストを生成することができる。

請求項４記載の発明は、ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックを選択しグループ化する手段により、任意の規模、個数のブロックのポートとポートの接続情報からなるコア（論理コア）を生成し（ステップ１）、チップの端子情報から、論理合成ツールで仮のチップのデザインを作成し、このデザインに前記チップの端子情報にある名称で端子を発生させ（ステップ２）、この内部に前記と同一デザインをセルとして発生させ（ステップ３）、前記デザインとセル間の同一名称のポートを接続し（ステップ４）、この接続ネットに対しチップの端子情報からデバイス・テクノロジに依存するＩ／Ｏバッファを挿入し（ステップ５）、ステップ１で生成したコア（論理コア）を前記セルと入れ替え（ステップ６）、このコア（論理コア）のトップの階層であるステップ２で作成したデザインの階層を展開してチップネットリストを生成する（ステップ７）、ことを特徴とする。

このように構成すれば、任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストから、チップネットリストを生成することができる。

請求の範囲第５項記載の発明は、任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストを生成する場合において、トップ階層のポート名をネット名から接続先のブロックのポート名と一致するようにポート名を修正し、ブロックのポートとポートの接続情報からなるネットリストを生成することを特徴とする。

請求の範囲第５項記載の発明は、ＡＳＩＣとプログラマブル・ロジックデバイ

スのコンカレント開発において、ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックをグループ化し、論理合成ツールの機能を使用してプログラマブル・ロジックデバイスのコア（論理コア）として切り出す場合、通常切り出したコア（論理コア）のトップのポート名称は、当該ポートに接続されているネット名となる。

これを論理合成ツールの機能を制御してポートからコア（論理コア）内部へネットをトレースし、最初に発見したブロックのポート名称に付け替えることを特徴とする論理合成ツールの制御方法である。

従って、本請求の範囲第５項のプログラマブル・ロジックデバイスのコア（論理コア）を生成する手法を論理合成ツールに実行させるプログラムを記憶した記憶媒体によれば、ポートの機能を判断することが難しいネット名がポートの名称となる場合の論理検証等デバッグの効率の低下を防ぐことができる。

上述のようにして、設計ドキュメントのデータからコア（論理コア）を生成し、該コア（論理コア）から階層構造及び接続情報を保持した状態でプログラマブル・ロジックデバイス用に新たにコア（論理コア）を生成することで、回路アーキテクチャを共有することができ、回路データを填め込み、機能検証したインスタンス内のデバイス・テクノロジに依存しない回路データ及びインスタンス間のネットは、ＡＳＩＣ化した場合に再検証を回避することが可能となり、ＡＳＩＣとプログラマブル・ロジックデバイス間の違いによる再設計も回避することが可能となる。

また、この発明において、ブロックのポートとポート間の接続情報からなるＡＳＩＣのコア（論理コア）から、ＡＳＩＣを構成する任意のブロックを選択し、グループ化することにより、選択したブロックで構成されるブロックのポートとポート間の接続情報からなるプログラマブル・ロジックデバイスのコア（論理コア）の生成を制御する論理合成ツールの制御装置において、論理合成ツールが生成した設計者が選択したブロックで構成されるプログラマブル・ロジックデバイスのコア（論理コア）のポートに対して、論理合成ツールがポート名として付与

したネット名を、該ポートにつながるネットを辿り、接続されているプログラマブル・ロジックデバイスを構成するブロックのポートの名称に変更する制御を行なう手段と、チップの端子情報から、チップの端子情報で指定されたポートをもつ仮のチップのデザインを論理合成ツールで生成する手段と、このデザイン内部に前記仮のチップのデザインをセルとして発生させる手段と、前記仮のチップのデザインとセル間の同一名称のポート間を接続し、この接続したネットに対し、チップの端子情報からデバイス・テクノロジに依存するＩ／Ｏバッファを挿入し、前記名称を変更した論理合成ツールが生成した、設計者が選択したブロックで構成されるプログラマブル・ロジックデバイスのコア（論理コア）とセルとを入れ替える手段と、このコア（論理コア）のトップ階層を展開することによりプログラマブル・ロジックデバイスのチップネットリストを生成する手段、とを具備することを特徴とする。

このように構成すれば、プログラマブル・ロジックデバイスのチップネットリストを生成することが可能となる。

また、本発明は、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発方法であって、ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、該グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成するネットリスト生成工程と、前記ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成する論理合成工程と、前記グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを前記ネットリスト生成工程により生成されたネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成するＲＯＭデータ生成工程と、前記論理合成工程により作成されたＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を前記ＲＯＭデータ生成工程による実機評価用ＲＯＭデータの生成とコンカレントに行なうレイアウ

ト作成工程と、前記ＲＯＭデータ生成工程により生成されたＲＯＭデータを用いた実機評価結果に基づく回路データの変更を前記ＡＳＩＣのレイアウト作成およびタイミング検証に反映する差分反映工程と、を含んだことを特徴とする。

また、本発明は、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発プログラムであって、ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、該グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成するネットリスト生成手順と、前記ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成する論理合成手順と、前記グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを前記ネットリスト生成手順により生成されたネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成するＲＯＭデータ生成手順と、前記論理合成手順により作成されたＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を前記ＲＯＭデータ生成手順による実機評価用ＲＯＭデータの生成とコンカレントに行なうレイアウト作成手順と、前記ＲＯＭデータ生成手順により生成されたＲＯＭデータを用いた実機評価結果に基づく回路データの変更を前記ＡＳＩＣのレイアウト作成およびタイミング検証に反映する差分反映手順と、をコンピュータに実行させることを特徴とする。

かかる発明によれば、ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成し、ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成し、グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを生成したネットリストに填め込んでプログラマブル・ロジッ

クデバイスの回路を記録した実機評価用ＲＯＭデータを生成し、作成したＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を実機評価用ＲＯＭデータの生成とコンカレントに行ない、生成したＲＯＭデータを用いた実機評価結果に基づく回路データの変更をＡＳＩＣのレイアウト作成およびタイミング検証に反映することとしたので、ＡＳＩＣとプログラマブル・ロジックデバイスの効率的なコンカレント開発が可能になり、ＡＳＩＣの開発期間を短縮することができる。

また、本発明は、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システムであって、ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、該グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成するネットリスト生成手段と、前記ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成する論理合成手段と、前記グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを前記ネットリスト生成手段により生成されたネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成するＲＯＭデータ生成手段と、前記論理合成手段により作成されたＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を前記ＲＯＭデータ生成手段による実機評価用ＲＯＭデータの生成とコンカレントに行なうレイアウト作成手段と、を備えたことを特徴とする。

この発明によれば、ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成し、ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成し、グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用

論理合成データを生成したネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成し、作成したＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を実機評価用ＲＯＭデータの生成とコンカレントに行なうこととしたので、ＡＳＩＣとプログラマブル・ロジックデバイスの効率的なコンカレント開発が可能になり、ＡＳＩＣの開発期間を短縮することができる。

また、本発明は、ネットワークに接続されたコンピュータからユーザが利用するＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システムにおいて、前記ユーザの要求に応じてＡＳＩＣの論理合成を実行するＡＳＩＣ論理合成手段と、前記ＡＳＩＣ論理合成手段により作成されたＡＳＩＣの論理合成結果が前記ユーザの要求する速度性能を満足したか否かを判断するＡＳＩＣ論理合成結果判断手段と、前記ＡＳＩＣ論理合成結果判断手段による判断結果に基づいてプログラマブル・ロジックデバイスの論理合成を実行するプログラマブル・ロジックデバイス論理合成手段と、前記ＡＳＩＣ論理合成手段によるＡＳＩＣ論理合成の実行結果及び前記プログラマブル・ロジックデバイス論理合成手段によるプログラマブル・ロジックデバイス論理合成の実行結果を前記コンピュータに表示する論理合成結果表示手段と、前記ＡＳＩＣ論理合成手段によるＡＳＩＣ論理合成の実行開始と実行結果及び前記プログラマブル・ロジックデバイス論理合成手段によるプログラマブル・ロジックデバイス論理合成の実行開始と実行結果を前記ユーザに電子メールで通知する論理合成通知手段と、を備えたことを特徴とする。

また、本発明は、ネットワークに接続されたコンピュータからユーザが利用するＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発方法において、前記ユーザの要求に応じてＡＳＩＣの論理合成を実行するＡＳＩＣ論理合成工程と、前記ＡＳＩＣ論理合成工程により作成されたＡＳＩＣの論理合成結果が前記ユーザの要求する速度性能を満足したか否かを判断するＡＳＩＣ論理合成結果判断工程と、前記ＡＳＩＣ論理合成結果判断工程による判断結果に基づいてプ

ログラマブル・ロジックデバイスの論理合成を実行するプログラマブル・ロジックデバイス論理合成工程と、前記ＡＳＩＣ論理合成工程によるＡＳＩＣ論理合成の実行結果及び前記プログラマブル・ロジックデバイス論理合成工程によるプログラマブル・ロジックデバイス論理合成の実行結果を前記コンピュータに表示する論理合成結果表示工程と、前記ＡＳＩＣ論理合成工程によるＡＳＩＣ論理合成の実行開始と実行結果及び前記プログラマブル・ロジックデバイス論理合成工程によるプログラマブル・ロジックデバイス論理合成の実行開始と実行結果を前記ユーザに電子メールで通知する論理合成通知工程と、を含んだことを特徴とする。

かかる発明によれば、ユーザの要求に応じてＡＳＩＣの論理合成を実行し、作成したＡＳＩＣの論理合成結果がユーザの要求する速度性能を満足したか否かを判断し、判断結果に基づいてプログラマブル・ロジックデバイスの論理合成を実行し、ＡＳＩＣ論理合成の実行結果及びプログラマブル・ロジックデバイス論理合成の実行結果をコンピュータに表示し、ＡＳＩＣ論理合成の実行開始と実行結果及びプログラマブル・ロジックデバイス論理合成の実行開始と実行結果をユーザに電子メールで通知することとしたので、ユーザは、論理合成の専任者を設置する必要がなく、いつでも論理合成が実行でき、専任者が実行したが如く論理合成品質を均一に保つことができるとともに、電子メールで論理合成の開始と結果の通知を受けることができ、論理合成の進捗を定期的にコンピュータで確認する必要がなくなる。

また、本発明は、前記ユーザの要求に応じて前記ＡＳＩＣを構成する機能ブロックから前記ユーザが指定する複数の機能ブロックのポート間接続情報から成るネットリストを生成するネットリスト生成手段と、前記ネットリスト生成手段により生成されたネットリストに論理合成済み対象機能ブロックのデータを填め込んでプログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを生成するＲＯＭデータ生成手段と、前記ＲＯＭデータ生成手段により生成されたＲＯＭデータの生成結果を前記コンピュータに表示するＲＯＭデータ生成結果表示手段

と、前記ＲＯＭデータ生成手段により生成されたＲＯＭデータの生成結果を前記ユーザに電子メールで通知するＲＯＭデータ生成結果通知手段とをさらに備えたことを特徴とする。

この発明によれば、ユーザの要求に応じてＡＳＩＣを構成する機能ブロックからユーザが指定する複数の機能ブロックのポート間接続情報から成るネットリストを生成し、生成したネットリストに論理合成済み対象機能ブロックのデータを填め込んでプログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを生成し、生成したＲＯＭデータの生成結果をコンピュータに表示するとともに、ユーザに電子メールで通知することとしたので、ユーザは、プログラマブル・ロジックデバイス専用の開発環境が必要なく、プログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを作成する負荷と、時間と、かかるコストとを削減することができる。

また、本発明は、前記ユーザが指定した前記ＡＳＩＣを構成する機能ブロックの設計が未完了で回路データが存在しない場合に、当該機能ブロックの入力端子及び出力端子に仮のフリップ・フロップなどを用いた回路を挿入したネットリストを生成する仮ネットリスト生成手段をさらに備えたことを特徴とする。

この発明によれば、ユーザが指定したＡＳＩＣを構成する機能ブロックの設計が未完了で回路データが存在しない場合に、当該機能ブロックの入力端子及び出力端子に仮のフリップ・フロップなどを用いた回路を挿入したネットリストを生成することとしたので、プログラマブル・ロジックデバイスのプロトタイピングによる検証において、検証対象ではない機能ブロックの設計が完了していなくとも、プロトタイピングによる検証を進めることができ、検証の効率化を図ることができる。

また、本発明は、前記ユーザが持つ最新の回路データとインプリメント設計者がインプリメント設計に取り込んでいる回路データ間の変更の規模を監視する監視手段と、前記監視手段による監視結果及びレイアウト設計にかかる時間に基づいて計画された日時に到達すると前記変更を前記ＡＳＩＣのインプリメント設計

への反映のタイミングにあることを前記ユーザ及び前記ＡＳＩＣのインプリメント設計者に電子メールで通知する変更タイミング通知手段と、前記変更タイミング通知手段に応答して前記ユーザが前記反映の日付を変更することで停止を要求する反映停止要求手段とをさらに備えたことを特徴とする。

この発明によれば、ユーザが持つ最新の回路データとインプリメント設計者がインプリメント設計に取り込んでいる回路データ間の変更の規模を監視し、監視結果及びレイアウト設計にかかる時間に基づいて計画された日時に到達すると変更をＡＳＩＣのインプリメント設計へ反映するタイミングにあることをユーザ及びＡＳＩＣのインプリメント設計者に電子メールで通知し、この通知に応答してユーザが反映の日付を変更することで停止を要求することとしたので、発生した変更を効率よくＡＳＩＣのレイアウト設計に反映することができるとともに、変更を反映するタイミングを設定することで、ユーザはいつまで変更することが可能かを判断することができ、早い段階でスケジュールの見直しができる。

また、本発明は、一方では、複数の機能ブロックの全部又は一部の機能ブロックを含むＦＰＧＡの端子情報が記述された第１デザインと、該第１デザインの下位層として記述された該ＦＰＧＡと同様の端子情報が記述された第２デザインとから、該第１デザイン、第２デザインにおける同一端子間が接続され、かつ、該端子間にＦＰＧＡ対応のバッファが挿入されたＦＰＧＡ用デザイン情報を生成し、他方では、該複数の機能ブロックを含むＡＳＩＣの端子情報が記述された第３デザインと、該第３デザインの下位層として記述された該ＡＳＩＣと同様の端子情報が記述された第４デザインとから、該第３デザイン、第４デザイン間における同一端子間が接続され、かつ、該端子間にＡＳＩＣ対応のバッファが挿入されたＡＳＩＣ用デザイン情報を生成し、前記第２デザイン、第４デザインのそれぞれを、各デザインが含む機能ブロックの接続情報に基づいて生成された回路情報で置き換える、ことでＦＰＧＡ及びＡＳＩＣのネットリストを生成することを特徴とする。

この発明によれば、一方では、複数の機能ブロックの全部又は一部の機能ブロ

ックを含むＦＰＧＡの端子情報が記述された第１デザインと、第１デザインの下位層として記述されたＦＰＧＡと同様の端子情報が記述された第２デザインとから、第１デザイン、第２デザインにおける同一端子間が接続され、かつ、端子間にＦＰＧＡ対応のバッファが挿入されたＦＰＧＡ用デザイン情報を生成し、他方では、複数の機能ブロックを含むＡＳＩＣの端子情報が記述された第３デザインと、第３デザインの下位層として記述されたＡＳＩＣと同様の端子情報が記述された第４デザインとから、第３デザイン、第４デザイン間における同一端子間が接続され、かつ、端子間にＡＳＩＣ対応のバッファが挿入されたＡＳＩＣ用デザイン情報を生成し、第２デザイン、第４デザインのそれぞれを、各デザインが含む機能ブロックの接続情報に基づいて生成された回路情報で置き換える、ことでＦＰＧＡ及びＡＳＩＣのネットリストを生成することとしたので、ＡＳＩＣとプログラマブル・ロジックデバイスの効率的なコンカレント開発が可能になり、ＡＳＩＣの開発期間を短縮することができる。

図面の簡単な説明

第１図は、本発明方法の原理を示すフローチャートであり、第２図は、本発明の実施の形態１の動作説明図であり、第３図は、本発明の論理合成ツール制御装置の一実施の形態例を示すブロック図であり、第４図は、制御装置の一実施の形態例を示すブロック図であり、第５図は、コア（論理コア）生成プログラムを示すフローチャート（１）であり、第６図は、コア（論理コア）生成プログラムを示すフローチャート（２）であり、第７図は、本発明の実施の形態１の詳細手順を示すフローチャートであり、第８図は、コア（論理コア）デザインチェック制御を示すフローチャートであり、第９図は、グループ化制御を示すフローチャートであり、第１０図は、Ｉ／Ｏバッファ挿入制御を示すフローチャートと処理のイメージを示す図であり、第１１図は、集積回路設計の手順を示すフローチャートであり、第１２図は、本実施の形態２に係るＡＳＩＣとＦＰＧＡのコンカレント開発の概念を説明するための説明図であり、第１３図は、本実施の形態２に係

るＡＳＩＣとＦＰＧＡのコンカレント開発システムのシステム構成を示す機能ブロック図であり、第１４図は、ファイルサーバ内にデータを格納するためのディレクトリ構成の一例を示す図であり、第１５図は、論理ＣＯＲＥ生成用テーブルの一例を示す図であり、第１６図は、本実施の形態２のＡＳＩＣとＦＰＧＡのコンカレント開発システムの処理手順を示すフローチャートであり、第１７図は、ログイン画面の一例を示す図であり、第１８図は、手順画面の一例を示す図であり、第１９図は、論理ＣＯＲＥ生成インタフェースプログラムの処理手順を示すフローチャートであり、第２０図は、論理ＣＯＲＥ生成インタフェース画面の一例を示す図であり、第２１図は、ＡＳＩＣ論理ＣＯＲＥ生成プログラムの処理手順を示すフローチャート（１）であり、第２２図は、ＡＳＩＣ論理ＣＯＲＥ生成プログラムの処理手順を示すフローチャート（２）であり、第２３図は、論理ＣＯＲＥチェックプログラムの処理手順を示すフローチャートであり、第２４図は、状況表示選択画面の一例を示す図であり、第２５図は、論理ＣＯＲＥ生成状況画面の一例を示す図であり、第２６図は、ＡＳＩＣの論理ＣＯＲＥ生成状況表示画面での処理手順を示すフローチャートであり、第２７図は、ＦＰＧＡ論理ＣＯＲＥ生成インタフェース画面の一例を示す図であり、第２８図は、ＦＰＧＡ論理ＣＯＲＥ生成プログラムの処理手順を示すフローチャート（１）であり、第２９図は、ＦＰＧＡ論理ＣＯＲＥ生成プログラムの処理手順を示すフローチャート（２）であり、第３０図は、互換パッケージテーブルの一例を示す図であり、第３１図は、論理合成インタフェースプログラムの処理手順を示すフローチャートであり、第３２図は、論理合成インタフェース画面の一例を示す図であり、第３３図は、ＦＰＧＡ論理合成インタフェース処理手順を示すフローチャートであり、第３４図は、論理合成状況表示画面の一例を示す図であり、第３５図は、フィッティングデータの投入と実行画面の一例を示す図であり、第３６図は、ＲＯＭデータ生成状況画面の一例を示す図であり、第３７図は、予定と実績画面の一例を示す図であり、第３８図は、手作業時間の一例を示す図である。

発明を実施するための最良の形態

以下、図面を参照して本発明の実施の形態例を詳細に説明する

実施の形態１．

第２図は本発明の実施の形態１の動作説明図である。この図は、回路アーキテクチャ検討結果を入力とし、設計データを共通化し、回路アーキテクチャを共有しつつ集積回路を開発するものである。

実現する機能をＨＤＬを手段として設計する場合、製品自身であるチップは、ある機能を持つインスタンスで構成される。このインスタンスは、ＨＤＬで設計された複数の機能で構成されるブロックを参照するものであり、特定のブロックがチップの機能を実現する上で必要であれば、該ブロックを複数参照するものである。

第２図に従い本発明の流れを説明する。Ｘ、Ｙ、Ｚは、回路アーキテクチャ検討結果から作成されるブロックのポート仕様を、定義済みのフォーマットに従って定義したテーブルである。これらテーブルＸ、Ｙ、Ｚは、ブロックを設計する際、必ず作成されるものであり、ブロック名、インスタンス名、ポート名、レンジ、入出力、種別、出力元インスタンス名、出力元インスタンスの出力ポート名等からなる。これらデータは、ユーザがマニュアルで入力するものである。このように、各テーブルを予め作成しておくことが本発明のポイントである。

Ｓ１は、前述の全インスタンスのテーブルデータを入力とし、インスタンス間及びインスタンスとデバイスパッケージの端子として定義された外部端子間が矛盾なく接続されているかをチェックする手段と、インスタンス間のネットを定義する手段とによりブロックのポートとブロック間接続情報のみからなるコア（論理コア）を生成するステップである。ステップＳ１でエラーがあった場合、エラーをユーザに返し、ユーザはデータ入力をやり直す（図中に示す円状の矢印１は繰り返しを示している）。

Ｓ２はＦＰＧＡテーブル２を参照し、ステップＳ１出力のコア（論理コア）を論理合成ツールに読み込み、プログラマブル・ロジックデバイスとするインスタ

ンスのグループを論理合成ツールの機能を制御して選択し、階層情報を維持した状態で新たなコア（論理コア）を生成し、デバイス・テクノロジに依存するＩ／Ｏバッファ等を含まないステップＳ１出力と同様のコア（論理コア）のネットリストを出力するステップである。この図で、Ｘ、Ｙはユーザが指定するものである。

Ｓ３は、定義済みフォーマットであるデバイスパッケージの端子仕様テーブルデータの端子名から仮のコア（論理コア）を生成し、テーブルデータで指定のデバイス・テクノロジに依存するＩ／Ｏバッファを仮のコア（論理コア）のポートに挿入後、ステップＳ２出力のネットリストと入れ替えるステップであり、本発明のポイントである。この処理によって対象とするデバイスのチップのネットリストができた後、論理合成ツールの機能を制御して対象のデバイス・テクノロジで論理合成済みの回路データをＦＰＧＡ合成結果のライブラリ３から読み込み、対応するブロックに填め込み、所望のデバイスのネットリストを完成させるものである。ＦＰＧＡとして生成したネットリストは、ＦＰＧＡのレイアウトツールでフィッティングし、ＲＯＭデータ化される。

また、ステップＳ２から、ＡＳＩＣ合成結果ライブラリ４を参照することによってＡＳＩＣ化している。

この実施の形態１によれば、集積回路開発において、ＲＴＬ設計の入力となるブロックのポート仕様の品質を事前に確保できる効果があり、更に機能ブロックが多く存在し、設計リソースも多い大規模集積回路の開発においても、事前にブロック間の接続が確認できていることにより、必ずチップが組み上がることを保証することができる。

第３図は本発明の論理合成ツール制御装置の一実施の形態例を示すブロック図である。図において、２２は全体の動作を制御する制御装置、２４は各種情報を表示するＣＲＴ、２１は制御装置２２に各種のコマンド等を入力する入力装置、２３は制御装置２２と接続され、各種の情報を記憶する記憶装置である。

入力装置２１は、コア（論理コア）生成プログラムの起動コマンド、論理合成

ツール制御コマンドを入力するものであり、記憶装置２３には論理合成ツール、コア（論理コア）生成プログラム、論理合成制御プログラムが記憶されている。

入力装置２１からブロックのテーブルファイルを指定してコア（論理コア）生成プログラム起動コマンドを入力すると、コア（論理コア）生成プログラムはテーブルファイルを読み込み、コア（論理コア）を生成し、記憶装置２３にファイル出力する。この処理中にエラーがあれば、そのエラー情報を表示装置であるＣＲＴ２４に表示する。

設計者は、エラーがあった場合、テーブルファイルを修正後、再度コマンドを実行する。論理合成ツールの制御では、先ず制御用の必要なデータをファイルとして用意し、コア（論理コア）生成プログラムで生成したコア（論理コア）ファイルと一緒に記憶装置２３内の所定の場所に格納する。そして、論理合成ツールの制御コマンドを入力装置２１から入力すると、論理合成ツールが起動し、処理結果がＣＲＴ２４に表示される。途中、処理に失敗した場合は、失敗した時の状態がＣＲＴ２４に表示される。

第４図は第３図の制御装置２２の一実施の形態例を示すブロック図である。第３図と同一のものは、同一の符号を付して示す。図において、３１は全体の動作を制御するＣＰＵ、３２は各種の情報を記憶するメモリ、２１は各種のコマンド等を入力するキーボード、２４は表示装置としてのＣＲＴである。２３は記憶装置であり、コア（論理コア）生成プログラム３６、論理合成ツール制御プログラム３５、論理合成ツール３４及びオペレーティングシステム（ＯＳ）３３より構成されている。３７は各構成要素間を接続するバスである。記憶装置２３としては、例えばハードディスク装置が用いられる。

このように構成されたシステムにおいて、ＣＰＵ３１はキーボード２１からのコマンドを入力すると、記憶装置２３を検索して、該当するプログラムを呼び出し、該当するプログラムを実行する。

第５図、第６図はコア（論理コア）生成プログラムを示すフローチャートである。ここでは、第２図のＸ、Ｙ、Ｚのテーブルデータを使用して説明する。テー

ブルデータX、Y、Z（インスタンスもX、Y、Z）のファイルを読み込み、1ファイルずつ処理を行なう。F1でXのデータを読み込み、F2でポートAのテーブルの全データを抽出してメモリに記憶し、このメモリのデータを参照しながら次の処理を行なう。F2'でポートは出力であるか否かをチェックする。

ポートAは入力であるので、インスタンス名とポート名をファイルAに出力するステップF3はスキップし、F3'に進む。F3'では、パッケージ端子との接続キーワードがあるか否かをチェックする。次に、パッケージ端子の接続キーワードはないため、パッケージ端子情報をメモリに格納するF4はスキップし、次のポートBの全データをメモリに記憶する。

ポートBは出力であるので、F3のステップにより“XB”を記憶装置23のファイルAに出力する。ここで、Xはファイル名、Bはポート名である。次に、パッケージ端子の接続キーワードはないためF4をスキップして次のポートZの全データをメモリに記憶する。ポートZは出力であるので、F3のステップにより“XZ［2：0］”を前記ファイルAに出力する。ここで、［2：0］は、2、1、0を示す。

次に、パッケージ端子の接続キーワードはないため、F4をスキップし、次のポートIの全データをメモリに記憶する。ポートIは入力であるのでF3をスキップし、次にパッケージ端子の接続キーワード“IO”があるため、“INPUT，in，2：0”をコア（論理コア）のポート名としてメモリに記憶する。次に、全ポート終了であるかどうかチェックする（F4'）。全ポートが終了していない場合には、ステップF2に戻る。

これで、Xの全ポートの処理が終了し、VHDLであれば、以下のような情報を持ったエンティティファイルを記憶装置23に出力する。

```
A：in　std_logic；
B：out　std_logic；
Z：out　std_logic　vector（2　downto 0）；
I：in　std_logic；
```

上記の処理をＹ，Ｚのデータについても行ない、全ファイルが終了すると（Ｆ５’）、Ｆ６のステップによりＶＨＤＬであれば、以下のような情報を持ったコア（論理コア）のエンティティファイルを記憶装置２３に出力する。

ＩＮＰＵＴ：ｉｎ　ｓｔｄ＿ｌｏｇｉｃ；

ＯＵＴ　：ｏｕｔ　ｓｔｄ＿ｌｏｇｉｃ；

また、ファイルＡには以下のようなデータが記録されている。

ＸＢ，ＸＺ［２：０］，ＹＣ［１：０］，ＹＯ，ＺＦ

次に、Ｆ７のステップで上記ファイルＡのデータを読み込み、Ｆ８のステップでメモリに記憶し、１ファイルずつ処理を行なう。Ｆ８’のステップでは、ポートは入力であるかどうかチェックする。ここで、Ｘのテーブルデータを例に説明する。ポートＡは入力であるので、Ｆ９のステップにより出力元のＹとポート名のＣを抽出し、続くＦ１０のステップによりＹＣ［１：０］をＦ８で記憶したデータで検索する。そして、Ｆ１０’で一致するかどうかチェックする。

検索の結果、存在するので、Ｆ１２のステップにより以下のような接続情報をファイルＢに出力する。結果が一致しない場合は、ポートの名称又はレンジが異なっている等のエラー情報を記憶装置２３のログファイルに出力すると共に、ＣＲＴ２４にも表示する。

Ａ＝＞ＹＣ

次に、同様にしてポートＢは出力であるので、Ｆ１１のステップにより、以下のような接続情報をファイルＢに出力する。

Ｂ＝＞ＸＢ

次に、同様にしてポートＺは出力であるので、Ｆ１１のステップにより、以下のような接続情報をファイルＢに出力する。

Ｚ＝＞ＸＺ

次に、ポートＩは入力であるが、Ｆ１０のステップの検索で“ＩＯＩＮＰＵＴ”が発見できず、Ｆ１３のステップでエラーとしてログファイルに出力されると共に、ＣＲＴ２４にも表示する。但し、パッケージの端子と接続されるポートの

場合は問題ではなく、逆にログに出力されたエラー情報からどのインスタンスのどのポートがパッケージのどの端子と接続したかの確認ができる。次に、Ｆ１２のステップにより、以下のような接続情報をファイルＢに出力する。

Ｉ＝＞ＩＯＩＮＰＵＴ

ステップＦ１２’では、全テーブルの全ポートが終了したかどうかをチェックし、終了していない場合には、ステップＦ８’に戻る。

以上の処理をＹ，Ｚのテーブルデータについても行ない、終了すると、Ｆ１４のステップにより、接続情報が作成できなかったポートをログファイルに出力し、Ｆ１５のステップによりＦ６のステップで出力したコア（論理コア）のエンティティリストとＦ１１とＦ１２のステップで出力した接続情報を結合し、コア（論理コア）のネットリストを生成し、記憶装置２３に出力してプログラムを終了する。

次に、図を用いて本発明の実施の形態１の詳細手順について説明する。第７図は本発明の実施の形態１の詳細手順を示すフローチャートである。このフローチャートは、第２図で説明した実施の形態１のうち、Ｓ１で出力したコア（論理コア）を入力とするＳ２以降の処理を詳細に示すものである。図中、実線は処理の流れを示し、破線はデータの流れを示す。処理の実行前に、入力装置を操作してステップＳ２で出力したコア（論理コア）と必要があれば、以降の説明で出てくる制御ファイルを制御プログラムが参照する記憶装置内の場所へ複写等の手段によって準備する。

先ず、コア（論理コア）デザインをチェックする（ａ１）。そして、該コア（論理コア）デザインがＯＫかどうかチェックする（ａ１’）。ＯＫでない場合には、第２図説明のテーブルデータをチェック後、再実行する（ａ１０）。ＯＫである場合、ファイル１２、１３を参照してグループ化し（ａ２）、コア（論理コア）１０として格納する。次に、端子名とバッファ名からなるテーブル１４を参照してダミーコア（論理コア）を生成する（ａ３）。この結果、ダミーコア（論理コア）１１が生成される。

次に、このダミーコア（論理コア）１１及びテーブル１４を基にしてＩ／Ｏバッファを挿入する（ａ４）。次に、コア（論理コア）１０とダミーコア（論理コア）１１とを比較チェックする（ａ５）。そして、ＯＫであるかどうかチェックする（ａ５’）比較の結果、ＯＫである場合には終端処理を行なう（ａ６）。ＯＫでない場合には、デバイスパッケージの端子仕様をチェックし（ａ９）、第２図説明のテーブルデータをチェック後、再実行する（ａ１０）。

ステップａ６の終端処理が終了したら、合成結果ファイル１５を参照して回路データを填め込み（ａ７）、ファイル１６を参照してＤＦＴ回路を挿入する（ａ８）。

次に、第７図で説明した各ステップの更に詳細な説明をする。第８図はコア（論理コア）デザインチェック制御を示すフローチャートである。先ず、コア（論理コア）のファイルがあるかどうかチェックし（ａ１１）、ある場合には、論理合成ツールで読み込ませる（ａ１２）。ない場合には、何もしない。ステップａ１（第７図参照）では、ステップＳ２で出力したコア（論理コア）を入力とし、コア（論理コア）のＨＤＬ記述文法、空き入力ポート等をチェックするステップで、コア（論理コア）のファイルが存在すれば、ステップａ１２で論理合成ツールにコア（論理コア）を読み込ませる。論理合成ツールは、読み込んだ際、文法等のチェックを行なう機能を持っており、結果がエラーであればＣＲＴで内容を確認し、ステップａ１０に示す通りブロックのポート仕様を記述したテーブルデータを見直し、再度デザインチェックを実行する。

第９図はグループ化制御を示すフローチャートで、第７図のステップａ２の詳細を示している。先ず、コア（論理コア）のファイルがあるかどうかチェックし（ａ１３）、ある場合にはグループ化するインスタンスのファイルがあるかどうかチェックする（ａ１４）。ある場合には、即ち、ステップａ１の結果に問題がない場合、プログラマブル・ロジックデバイス化する対象のブロック名を１行に１ブロックずつ記載されたファイル１３があればそれを読み込み（ａ２１）、対象のブロックの階層情報を維持した状態で新たなコア（論理コア）として論理合

成ツールのグルーピング機能を制御してグループ化する（ａ２２）。

グルーピング機能を使用する場合、コア（論理コア）のポート名がインスタンス間を接続しているネット名となるため、コア（論理コア）の全てのポートに対してコア（論理コア）内部へネットをトレースし、最初に発見したインスタンスのポート名でコア（論理コア）に新規ポートを発生させネットの名前となっているコア（論理コア）のポートに接続されているネットを発生させた新規ポートに接続し、ネット名となっているポートを削除する（ａ２３）。新しいポートを作成して古いポートを消すためである。

次に、モニタポート情報があるかないかチェックする（ａ２３’）。ある場合には、コア（論理コア）に指定のインスタンスのポート名でポートを発生させ、ブロックの端子と接続する（ａ２４）。ない場合には、デバイス・テクノロジに依存するＩ／Ｏバッファ等を含まないネットリストを記憶装置１０（第７図参照）に出力する（ａ２５）。この場合、表示装置においては、論理合成ツールの表示機能を使用して確認することができる。

このステップａ２では、他に必要があれば、グループ化したブロックのポートをコア（論理コア）のポートとして発生させる（ａ２４）こともできる。この機能は、プログラマブル・ロジックデバイスの機能評価時にモニタしたいポートがある場合に使用する。指定方法は、ファイル１２に示すように、ポート名を１行に１ポートずつ記載したファイルをグルーピングとポート名称変更後に読み込ませることで行なう。

ここまでは、設計者自身がブロックのポート仕様を決めるデータを用いてコア（論理コア）を生成するため、コア（論理コア）内のブロック間のポート接続に間違いはないはずである。しかしながら、ステップＳ１（第２図参照）で生成するコア（論理コア）をそのまま使用するＡＳＩＣの場合、コア（論理コア）内ブロックのテーブルデータに定義するデバイスパッケージの端子名称はデバイスパッケージの端子仕様を参照するのが普通であり、従って単純なミスやプリント基板設計からの変更等で相違がある可能性がある。これとは逆に、ステップａ２（

第７図参照）で生成したコア（論理コア）のポートがデバイスパッケージの端子となる場合、該ポートが仕様となるため、前述と同様に相違がある可能性がある。この問題をステップａ３からａ５のステップで解消している。

第１０図はＩ／Ｏバッファ挿入制御を示すフローチャートと処理のイメージを示す図である。このフローチャートは、第７図のステップａ４の処理を示している。以下の説明では、ａ３からａ５は別としているが、処理は１つのコマンドで一連の処理である。

先ず、外部端子情報及びコア（論理コア）のファイルがあるかどうかチェックする（ａ１５）。ある場合には、以下の処理を行なう。ａ３では、ａ２と違い第７図の１４で示すように第１カラムに端子名称、第２カラムに使用するデバイステクノロジに依存するＩ／Ｏバッファ名を定義したデバイスパッケージの端子仕様のテーブルデータが存在すれば、それを読み込ませ、ダミーのチップのデザイン（第１カラムの名称の端子情報を記述した第１デザイン）を作成し（ａ３１）、この内部に前記ダミーチップのデザインをセル（第１デザインの下位層として記述した第１デザインと同様の端子情報を記述した第２デザイン）として発生させ、前記デザインと前記セル間の同一名称のポートを接続し、記憶装置１１にダミーのコア（論理コア）を出力する。

ステップａ４では、記憶装置１１のダミーのコア（論理コア）を読み込み、デバイスパッケージの端子仕様テーブルデータ１４の第２カラムにあるＩ／Ｏバッファをダミーのコア（論理コア）のネットに対しチップの端子情報からデバイス・テクノロジに依存するＩ／Ｏバッファを挿入する（ａ３２）。

ステップａ５では、上記ダミーコア（論理コア）のポートのうち機能に関係のない論理に係わらないデバイス特有でテスト用等のポートを削除し、内部のセルとして記憶装置１０のコア（論理コア）と入れ替え（ａ３３）、端子名の不一致があるかどうかチェックする（ａ３３’）。２つのコア（論理コア）間のポート名が全て一致すれば、填め込みに成功である。こうすることで、デバイスパッケージの端子仕様テーブルデータの端子名称とコア（論理コア）のポート名称をク

ロスチェックでき、上記の問題を解消できる。ある場合は処理は終了し、ない場合にはネットリストを出力する。

失敗した場合には、ステップａ９に示す通り、デバイスパッケージの端子仕様をチェックし、ａ３のステップから再実行するか、ステップａ１０に示す通りブロックの端子仕様を定義したテーブルデータをチェックし、ステップＳ１（第２図参照）から再実行する。成功した場合、コア（論理コア）の階層を展開することでチップと化す。

ステップａ６では、前記チップの全ブロックの入出力ポートにフリップフロップをブロック内部に接続し、終端する。この終端処理は、仮に回路設計が完了していないブロックが存在したとしても問題なくネットリストを生成し、レイアウト作業を可能とするためのものである。

ステップａ７では、論理合成済みの回路データを合成結果のライブラリから読み込み、対応するブロックに填め込む。この時、チップ内部のブロックに対する合成結果が存在せず、且つ該ブロックのデザイン名称又はセル名称が名前付け規則に従ったものであれば、該ブロックの入力ポートと出力ポートを予め規則に従って該ブロック内部で接続処理する。

この処理は、プログラマブル・ロジックデバイスの場合には、存在せず、プログラマブル・ロジックデバイスのＤＬＬ等特有のブロックをＡＳＩＣとする場合に施す。このブロックを削除せずブロック内部で接続する処理によって、アーキテクチャが保持できる。また、ブロック内部にメモリ等のマクロが存在する場合は、デバイス特有のメモリの回路データに入れ替える。

ａ８はＡＳＩＣの場合の処理で、デバイスをテストするＳＣＡＮテスト回路の自動挿入、又はＳＣＡＮテスト回路を接続するブロックの順序を定義したファイル１６を入力し、定義順に接続する。

以上、述べたように、本実施の形態１によれば、設計ドキュメントからのデータからコア（論理コア）を生成し、このコア（論理コア）から階層構造及び接続情報を保持した状態でプログラマブル・ロジックデバイス用に新たにコア（論理

コア）を生成することによって、回路アーキテクチャを共有できる効果があり、このコア（論理コア）に対して回路データを填め込み機能を検証したインスタンス内のデバイス・テクノロジに依存しない回路データ及びインスタンス間のネットは、ＡＳＩＣ化した場合に再検証を回避することが可能となる。よって、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発を効率よく進めることを可能にできる効果もある。

実施の形態２．

次に、本発明に係るＡＳＩＣとＦＰＧＡのコンカレント開発システムについて詳細に説明する。なお、論理の設計言語には、Ｃ言語、ＵＭＬなどがあるが、本実施の形態２では、ＨＤＬを設計言語として説明する。

まず、本実施の形態２に係るＡＳＩＣとＦＰＧＡのコンカレント開発の概念について説明する。第１２図は、本実施の形態２に係るＡＳＩＣとＦＰＧＡのコンカレント開発の概念を説明するための説明図である。同図に示すように、このＡＳＩＣとＦＰＧＡのコンカレント開発の特徴は、ＦＰＧＡによるプロトタイピング検証をＡＳＩＣのインプリメント設計とコンカレントに進めるために、プロトタイピング検証に必要なＦＰＧＡの回路を記録したＲＯＭデータをＡＳＩＣのインプリメント設計から提供し、ＦＰＧＡとＡＳＩＣ開発をシームレスにできることである。

このＡＳＩＣとＦＰＧＡのコンカレント開発では、回路アーキテクチャ検討においてＦＰＧＡによるプロトタイピング検証を考慮し、適当な予想規模の実現回路での実現機能の分割、ＡＳＩＣとＦＰＧＡ間の構造的違いに係る機能の階層化などを行ない、ＡＳＩＣとＦＰＧＡ設計間の共通の機能ブロック構成及び機能ブロックのポート仕様を作成する。この機能ブロック構成及び機能ブロックのポート仕様データは、ＡＳＩＣのインプリメント設計のフロアプラン、論理合成、レイアウト設計の共通のデータとなる。

ＲＴＬ設計・検証では、回路アーキテクチャ検討結果の構造に従ってＡＳＩＣとＦＰＧＡ共通のＲＴＬ設計を行ない、機能毎にコーナーケースを重点に論理検

証ツールを用いてソフト検証を行なう。このＲＴＬ設計・検証とコンカレントにＡＳＩＣの論理合成を実施し、ＡＳＩＣとしての特性が確保できれば、次にＦＰＧＡの論理合成を実施し、逐次プロトタイピング用基板上のＦＰＧＡに回路を形成し、ＦＰＧＡのプロトタイピングによる検証を可能にする。

一方、ＡＳＩＣのレイアウトでは、ＡＳＩＣとＦＰＧＡ設計間の共通の機能ブロック構成を基に、ソフト検証とプロトタイピングによる検証結果を随時コンカレントに反映させることによって、ＦＰＧＡのプロトタイピングによる検証の完了からＡＳＩＣ開発完了までの期間を短縮する。

次に、本実施の形態２に係るＡＳＩＣとＦＰＧＡのコンカレント開発システムのシステム構成について説明する。第１３図は、本実施の形態２に係るＡＳＩＣとＦＰＧＡのコンカレント開発システムのシステム構成を示す機能ブロック図である。同図に示すように、このＡＳＩＣとＦＰＧＡのコンカレント開発システム２００は、ファイアウォール２１０と、Ｗｅｂサーバ２２０と、ユーザ認証サーバ２３０と、ユーザ管理サーバ２４０と、論理合成サーバ２５０と、メールサーバ２６０と、ファイルサーバ２７０と、アプリケーションサーバ２８０と、監視サーバ２９０とを有する。また、このＡＳＩＣとＦＰＧＡのコンカレント開発システム２００は、インターネットを介してＷｅｂクライアント１００から利用することができる。

ファイアウォール２１０は、インターネットからのアクセス要求に対し、設定された通信手順に従ったアクセス要求のみを受入れるコンピュータであり、ＡＳＩＣとＦＰＧＡのコンカレント開発システム２００への外部からの不正なアクセスを防ぐ。

Ｗｅｂサーバ２２０は、インターネットを通じて成されるＷｅｂクライアント１００からの要求に対して情報送信を行なうコンピュータであり、ＡＳＩＣ又はＦＰＧＡを構成する機能ブロックのポートと機能ブロックのポート間の接続情報のみから論理ＣＯＲＥを生成する論理ＣＯＲＥ生成インタフェースプログラム２２１、論理合成インタフェースプログラム２２２、フィッティングインタフェー

スプログラム２２３、状況表示インタフェースプログラム２２４、論理ＣＯＲＥ生成に必要なフォーマットファイルと論理合成に必要なフォーマットファイルとを有し、このコンピュータがＷｅｂクライアント１００からの要求に応じてこれらのプログラムを実行し、Ｗｅｂクライアント１００に結果を送信する。

ユーザ認証サーバ２３０は、ユーザの認証を行なうコンピュータであり、契約に基づいて使用するユーザ名とパスワードが登録されている。ユーザ管理サーバ２４０は、ユーザの登録、削除を行なうコンピュータであり、契約に基づいて使用するユーザ名と後述するＰｒｏｊｅｃｔ名とメールアドレスが登録されている。

論理合成サーバ２５０は、ＡＳＩＣ論理ＣＯＲＥ生成プログラム２５１と、ＦＰＧＡ論理ＣＯＲＥ生成プログラム２５２と、ＡＳＩＣ論理合成プログラム２５３と、ＦＰＧＡ論理合成プログラム２５４と、ＦＰＧＡのレイアウトであるフィッティングプログラム２５５とを有するコンピュータであり、この論理合成サーバ２５０にある論理合成及びフィッティングのプログラムは、論理合成インタフェースプログラム２２２から起動され、ファイルサーバ２７０に格納されているＲＴＬソースを読込み、ＡＳＩＣ及びＦＰＧＡの論理合成とＦＰＧＡのフィッティングを実行する。

メールサーバ２６０は、メール送受信ソフトウェアを有するコンピュータであり、Ｗｅｂサーバ２２０から実行された処理情報、Ｗｅｂクライアント１００からの情報をユーザとシステム内へメール配信する。

ファイルサーバ２７０は、論理合成対象ＲＴＬソース、論理合成結果、ＦＰＧＡのＲＯＭデータなどを格納する記憶媒体である。第１４図は、ファイルサーバ内にデータを格納するためのディレクトリ構成の一例を示す図である。同図に示すＰｒｏｊｅｃｔ４１とは、ＡＳＩＣ及びＦＰＧＡを開発するＰｒｏｊｅｃｔ名あるいはＡＳＩＣのニックネームのディレクトリであり、この名称は契約に基づいて設定される。

ＩＯ４２は、論理ＣＯＲＥを生成するためのデータを格納するディレクトリで

あり、ＡＳＩＣを構成する機能ブロック単位に、第１５図に示す機能ブロックのポート仕様を記載した論理ＣＯＲＥ生成用テーブルのファイルを格納するディレクトリＡＳＩＣ４８と、論理ＣＯＲＥ生成インタフェースプログラム２２１がユーザの指定に従ってディレクトリＡＳＩＣ４８内の論理ＣＯＲＥ生成用テーブルのファイルを複写して格納するディレクトリＦＰＧＡ４９とをもつ。また、ディレクトリＦＰＧＡ４９の下に、ユーザが指定するＦＰＧＡの番号の数のディレクトリが存在する。なお、論理ＣＯＲＥ生成用テーブル及び論理ＣＯＲＥ生成インタフェースプログラム２２１の詳細については後述する。

ＣＯＲＥ４３は、論理ＣＯＲＥ生成プログラムが生成した論理ＣＯＲＥを格納するディレクトリであり、ＡＳＩＣを構成する機能ブロックのポート仕様から生成したＡＳＩＣの論理ＣＯＲＥを格納するディレクトリＡＳＩＣ５０と、ＦＰＧＡを構成する機能ブロックのポート仕様からＦＰＧＡの論理ＣＯＲＥを格納するディレクトリＦＰＧＡ５１をもつ。

ＲＴＬ４４は、後述する論理合成インタフェースからユーザがアップロードするＨＤＬ（ハードウェア記述言語）で表現された回路設計データ（以降ＲＴＬ）を格納するディレクトリであり、ＡＳＩＣの論理ＣＯＲＥを構成する機能ブロック単位に格納するディレクトリＡＳＩＣ５２と、ユーザの指定に従ってＡＳＩＣディレクトリ内のＲＴＬを複写して格納するディレクトリＦＰＧＡ５３とをもつ。

ＳＹＮＴＨＥＳＩＳ４５は、論理合成サーバ２５０がＲＴＬを論理合成した結果を格納するディレクトリであり、ＡＳＩＣの論理合成結果をＡＳＩＣの論理ＣＯＲＥを構成する機能ブロック単位に格納するディレクトリをもつディレクトリＡＳＩＣ５４と、それぞれのＦＰＧＡの論理合成結果を格納するディレクトリをもつディレクトリＦＰＧＡ５５をもつ。

ＲＯＭ４６は、ＦＰＧＡの論理合成結果を基にＦＰＧＡのレイアウトであるフィッティング後生成されるＦＰＧＡの回路データを記憶するＲＯＭデータを格納するディレクトリであり、ＦＰＧＡ毎に格納するディレクトリをもつ。ＬＡＹＯ

ＵＴ４７は、ＡＳＩＣのレイアウト設計担当者がＡＳＩＣのレイアウト設計を行なう作業ディレクトリである。

アプリケーションサーバ２８０は、ＡＳＩＣのフロアプラン、レイアウト設計、タイミング検証を行なうプログラムを有するコンピュータである。このコンピュータでインプリメント設計者がＡＳＩＣのフロアプラン、レイアウト設計、タイミング検証を行なう。

監視サーバ２９０は、ユーザ管理サーバ２４０に登録されているユーザ名とＰｒｏｊｅｃｔ名を取得し、それぞれのＰｒｏｊｅｃｔ名のディレクトリ内にあるデータにおいて、インプリメント設計者が管理するディレクトリＬＡＹＯＵＴ４７内のデータとその他のディレクトリ内のデータとの比較、論理合成、レイアウト処理にツールが要した時間の収集を１日に１回ずつ２回に分けて行ない、１回目がインプリメント設計者への変更内容の通知を行ない、２回目に収集した時間を基に変更を反映するスケジュールを更新するコンピュータである。

次に、本実施の形態２のＡＳＩＣとＦＰＧＡのコンカレント開発システム２００の処理手順について説明する。第１６図は、本実施の形態２のＡＳＩＣとＦＰＧＡのコンカレント開発システム２００の処理手順を示すフローチャートである。

同図に示すように、このＡＳＩＣとＦＰＧＡのコンカレント開発システム２００にＷｅｂクライアント１００を用いてアクセスすると、Ｗｅｂサーバ２２０は、第１７図に示すログイン画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は、受取ったデータに基づいて画面を表示する。ユーザは、このログイン画面で契約に基づいて登録しているユーザ名とパスワードを入力してＬｏｇｉｎボタンを押すと、Ｗｅｂクライアント１００は、ユーザ名とパスワードをＷｅｂサーバ２２０に送る。

Ｗｅｂサーバ２２０は、受取ったユーザ名とパスワードをユーザ認証サーバ２３０に問い合わせる。ユーザ認証サーバ２３０は、ユーザ名とパスワードが登録されているか否かを確認し（ステップＳ５０１）、その結果をＷｅｂサーバ２２

０に返す。Ｗｅｂサーバ２２０は、ユーザ認証サーバ２３０から拒絶された場合、ログイン拒否画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は、受取ったデータに基づいてログイン拒否を画面に表示して終了する（ステップＳ５０２）。

Ｗｅｂサーバ２２０は、ユーザ認証サーバ２３０からログインを受け付けられた場合、第１８図に示す手順画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取ったデータに基づいて手順画面を表示する（ステップＳ５０３）。

ユーザは、第１８図に示した手順画面に従いメニューを選択し、ＡＳＩＣとＦＰＧＡのコンカレント開発を行なう。ユーザが「フォーマット１」及び「フォーマット２」を選択すると（ステップＳ５０４の肯定）、Ｗｅｂサーバ２２０内にある対象の設計に必要なフォーマットファイルがＷｅｂクライアント１００にダウンロードされる（ステップＳ５０５）。

「フォーマット１」は、第１５図に示した論理ＣＯＲＥ生成用テーブルのフォーマットデータである。「フォーマット２」は、後述するＡＳＩＣの論理合成で使用するＡＳＩＣのパッケージの端子名称と端子番号割り付け、及びＡＳＩＣと外部デバイス間を電気的にインタフェースするＩＯバッファを定義したテーブルのフォーマットデータである。

ユーザが論理ＣＯＲＥ生成を選択すると（ステップＳ５０６の肯定）、Ｗｅｂクライアント１００は論理ＣＯＲＥ生成が選択された事をＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は論理ＣＯＲＥ生成インタフェースプログラム２２１を起動する（ステップＳ５０７）。

ユーザが論理合成を選択すると（ステップＳ５０８の肯定）、Ｗｅｂクライアント１００は論理合成が選択された事をＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は論理合成インタフェースプログラム２２２を起動する（ステップＳ５０９）。

ユーザがフィッティングを選択すると（ステップＳ５１０の肯定）、Ｗｅｂク

ライアント１００はフィッティングが選択された事をＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０はフィッティングインタフェースプログラム２２３を起動する（ステップＳ５１１）。

ユーザが状況表示を選択すると（ステップＳ５１２の肯定）、Ｗｅｂクライアント１００は状況表示が選択された事をＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は状況表示インタフェースプログラム２２４を起動する（ステップＳ５１３）。

次に、論理ＣＯＲＥ生成インタフェースプログラム２２１の処理手順について説明する。第１９図は、論理ＣＯＲＥ生成インタフェースプログラム２２１の処理手順を示すフローチャートである。同図に示すように、この論理ＣＯＲＥ生成インタフェースプログラム２２１は、第２０図に示す論理ＣＯＲＥ生成インタフェース画面の表示制御データをＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は受取った論理ＣＯＲＥ生成インタフェース画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取った論理ＣＯＲＥ生成インタフェース画面の表示制御データに基づいて論理ＣＯＲＥ生成インタフェース画面を表示する（ステップＳ８０１）。

ユーザは、Ｐｒｏｊｅｃｔ名を入力して実行ボタンを選択し、Ｗｅｂクライアント１００によって表示されるファイル選択画面に従ってＡＳＩＣの論理ＣＯＲＥを生成するためのＡＳＩＣを構成する機能ブロックのＩＯ仕様を定義した第１５図に示す論理ＣＯＲＥ生成用テーブルのファイルを指定する。

Ｗｅｂクライアント１００は、Ｐｒｏｊｅｃｔの指定データとＩＯの論理ＣＯＲＥ生成用テーブルのファイルをＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は、ファイルサーバ２７０内に、第１４図に示したＩＯ４２，ＣＯＲＥ４３，ＲＴＬ４４，ＳＹＮＴＨＥＳＩＳ４５，ＲＯＭ４６，ＬＡＹＯＵＴ４７などのディレクトリを作成し（ステップＳ８０２）、ＩＯ４２下のディレクトリＡＳＩＣ４８に論理ＣＯＲＥ生成用テーブルのファイルを格納し、Ｐｒｏｊｅｃｔ名を指定し、ＡＳＩＣ論理ＣＯＲＥ生成プログラム２５１を実行し、実行プロセスＩＤ

をメモリに格納する（ステップＳ８０３）。

ここで、ＡＳＩＣ論理ＣＯＲＥ生成プログラム２５１の処理手順について説明する。第２１図及び第２２図は、ＡＳＩＣ論理ＣＯＲＥ生成プログラム２５１の処理手順を示すフローチャートである。第２１図に示すように、このＡＳＩＣ論理ＣＯＲＥ生成プログラム２５１は、第１５図に示した論理ＣＯＲＥ生成用テーブルX、Y、Zを、第１４図のディレクトリＡＳＩＣ４８から読込み、１ファイルずつ処理を行なう。

論理ＣＯＲＥ生成用テーブルX、Y、Zは、回路アーキテクチャ検討結果から作成される機能ブロックのポート仕様を、定義済みの「フォーマット１」に従って定義したテーブルである。このテーブルは、機能ブロックを設計する際必ず作成されるものであり、機能ブロック名、インスタンス名、ポート名、レンジ、入出力、種別、出力元インスタンス名、出力元インスタンスの出力ポート名などから成る。インスタンスとは、ＡＳＩＣを構成する機能ブロックを参照するものであり、同一機能の機能ブロックを複数使用する場合はこのインスタンスの名称を変えて組込まれる。

まず、Xのデータを読込み（ステップＳ１００１）、ポートAのテーブルの全データを抽出してメモリに格納し（ステップＳ１００２）、ポートAが出力か否かを調べる（ステップＳ１００３）。その結果、ポートAは入力であるので、次にパッケージ端子の接続キーワード”ＩＯ”の有無を調べる（ステップＳ１００５）。その結果、接続キーワード”ＩＯ”はないので、次のポートＢの全データをメモリに格納する。

ポートＢは出力であるので、”ＸＢ”を第１４図のディレクトリＡＳＩＣ５０にAの名称でファイル出力する（ステップＳ１００４）。次にパッケージ端子の接続キーワード”ＩＯ”はないので、次のポートZの全データをメモリに格納する。ポートZは出力であるので、”ＸＺ[2:0]”をファイルAに出力する。次にパッケージ端子の接続キーワード”ＩＯ”はないので、次のポートＩの全データをメモリに格納する。

ポートＩは入力であり、次にパッケージ端子の接続キーワード”ＩＯ”があるので”ＩＮＰＵＴ，in，２：０”を論理ＣＯＲＥのポート名としてメモリに格納する（ステップＳ１００６）。これでＸの全ポートの処理が終了し（ステップＳ１００７の肯定）、ＨＤＬの文法に従ってポート名と入出力とレンジ情報を定義したエンティティファイルを第１４図のディレクトリＡＳＩＣ５０にＸの名称でファイル出力する（ステップＳ１００８）。出力される内容は、ＶＨＤＬであれば以下のようなものである。

Ａ： in std_logic；

Ｂ： out std_logic；

Ｚ： out std_logic_vector（2 downto 0）；

Ｉ： in std_logic；

そして、上記の処理をＹ、Ｚのデータについても行なう。また、この段階でファイルＡには次のようなデータが記録されている。

ＸＢ，ＸＺ[２：０]，ＹＣ[１：０]，ＹＯ，ＺＦ

上記Ｘ、Ｙ、Ｚの全ファイルの処理が終了すると（ステップＳ１００９の肯定）、上記ファイルＡ（ステップＳ１０１０）のデータを読込み、メモリに格納し（ステップＳ１０１１）、１ファイルずつ処理を行なう。ここでは、Ｘのテーブルデータを例に説明する。まず、ポートＡが入力であるか否かを調べ（ステップＳ１０１２）、その結果、ポートＡは入力であるので、出力元のＹとポート名のＣを抽出する（ステップＳ１０１３）。そして、メモリをＹＣ[１：０]で検索し（ステップＳ１０１４）、一致するデータの有無を調べる（ステップＳ１０１６）。その結果、一致するデータがメモリに存在するので、以下のような接続情報を第１４図のディレクトリＡＳＩＣ５０にＢの名称でファイル出力する（ステップＳ１０１８）。結果が一致しない場合は、ポートの名称またはレンジが異なっているなどのエラー情報をディレクトリＡＳＩＣ５０のログファイルに出力する（ステップＳ１０１７）。

Ａ => ＹＣ

同様にして、ポートＢは出力であるので、以下のような接続情報をディレクトリＡＳＩＣ５０のファイルＢに出力する（ステップＳ１０１５）。

Ｂ => ＸＢ

同様にして、ポートＺは出力であるので、以下のような接続情報をディレクトリＡＳＩＣ５０のファイルＢに出力する。

Ｚ => ＸＺ

同様にして、ポートＩは入力であり、検索で”ＩＯＩＮＰＵＴ”が発見できないので、エラーとしてディレクトリＡＳＩＣ５０のログファイルに出力する。ただし、パッケージの端子と接続されるポートの場合は問題ではなく、逆にログに出力されたエラー情報からどのインスタンスのどのポートがパッケージのどの端子と接続したかの確認ができる。次に、以下のような接続情報をディレクトリＡＳＩＣ５０のファイルＢに出力する。

Ｉ=> ＩＯＩＮＰＵＴ

以上の処理をＹ,Ｚのテーブルデータについても行って終了すると（ステップＳ１０１９の肯定）、接続情報が作成できなかったポートをログファイルに出力し（ステップＳ１０２０）、ＶＨＤＬであれば、以下のような情報をもつたＡＳＩＣの論理ＣＯＲＥのエンティティファイルをディレクトリＡＳＩＣ５０に出力する（ステップＳ１０２１）。

ＩＮＰＵＴ： in std_logic；

ＯＵＴ　　： out std_logic；

また、論理ＣＯＲＥのエンティティファイルと接続情報を結合し、ＡＳＩＣの論理ＣＯＲＥをディレクトリＡＳＩＣ５０に第１７図で指定されたＰｒｏｊｅｃｔ名でファイル出力して処理を終了し、第１９図に示した論理ＣＯＲＥ生成インタフェースプログラム２２１に戻る（ステップＳ１０２２）。

論理ＣＯＲＥ生成インタフェースプログラム２２１は、ディレクトリＡＳＩＣ５０内のログファイルにエラーがないかを検索する。もしエラーが記録されていれば（ステップＳ８０４の肯定）、エラーを読み込み（ステップＳ８０５）、Ｐ

ｒｏｊｅｃｔ名と、ユーザ管理サーバ２４０にＰｒｏｊｅｃｔ名で問い合わせて得たユーザのメールアドレスと、エラー情報と、プロセスＩＤとをメールサーバ２６０へ送り、メールサーバ２６０からユーザへメールを送信する（ステップＳ８０６）。ユーザは、このメールでエラー内容を確認し、エラーがなくなるまで論理ＣＯＲＥ生成の処理を繰り返す。

一方、ログファイルにエラーが記録されていなければ（ステップＳ８０４の否定）、Ｐｒｏｊｅｃｔ名と、ユーザ管理サーバ２４０にＰｒｏｊｅｃｔ名で問い合わせて得たユーザのメールアドレスと、プロセスＩＤと、論理ＣＯＲＥ生成終了メッセージとをメールサーバ２６０へ送り（ステップＳ８０７）、メールサーバ２６０からユーザへメールを送信する。

また、ディレクトリＡＳＩＣ５０に”ＣＯＲＥ＋プロセスＩＤ＋日時”の名称でディレクトリを作成し、ディレクトリＡＳＩＣ５０にＰｒｏｊｅｃｔ名で生成した論理ＣＯＲＥとログファイルとをこのディレクトリに移動し、このディレクトリにＩＯの名称でディレクトリを作成し、論理ＣＯＲＥの生成に使用したディレクトリＡＳＩＣ４８にあるテーブルデータをこのＩＯディレクトリに移動し、論理合成サーバ２５０内の論理ＣＯＲＥチェックプログラムにこのＰｒｏｊｅｃｔ名と”ＣＯＲＥ＋プロセスＩＤ＋日時”の名称で作成した論理ＣＯＲＥのディレクトリ名を指定し、論理ＣＯＲＥチェックプログラムを起動する（ステップＳ８０８）。

この論理ＣＯＲＥチェックプログラムは、ＡＳＩＣの論理ＣＯＲＥの文法チェック、論理ＣＯＲＥを構成する各インスタンスの入出力ポートの未接続チェックを行ない、レポートを出力するように論理合成ツールのコマンドが記述されている。

ここで、論理ＣＯＲＥチェックプログラムの処理手順について説明する。第２３図は、論理ＣＯＲＥチェックプログラムの処理手順を示すフローチャートである。この論理ＣＯＲＥチェックプログラムは、受取ったＰｒｏｊｅｃｔ名のディレクトリ下にあり、第１４図のディレクトリＡＳＩＣ５０に対象とするＡＳＩＣ

の論理ＣＯＲＥのディレクトリがあるかを調べ（ステップＳ１２０１）、もしあれば、その論理ＣＯＲＥのディレクトリがあるＡＳＩＣディレクトリ内にＷＯＲＫディレクトリを作成し（ステップＳ１２０２）、論理合成ツールにチェックを実行させる（ステップＳ１２０３）。

実行が終了すると、ＷＯＲＫディレクトリ内にあるレポートファイル内からエラー情報を抽出し、ユーザ管理サーバ２４０にＰｒｏｊｅｃｔ名で問い合わせて得たユーザのメールアドレスと論理ＣＯＲＥのディレクトリ名から抽出したプロセスＩＤと共にメールサーバ２６０へ送り、メールサーバ２６０からユーザへメール送信する（ステップＳ１２０４）。ユーザはこの送られたメールの内容でエラー情報が意図したものか確認する。

この一連のフローで生成された論理ＣＯＲＥは、ユーザの論理設計においてＡＳＩＣを構成するインスタンス間の結線ミスが無い限り必ずＡＳＩＣとして組み上がることを保証するものである。この効果は、論理検証にて現れる。ＡＳＩＣを構成する複数のインスタンスの機能検証において期待と違った動作をした場合、インスタンス間の接続は保証できていることからインスタンスを構成する各機能に絞ったデバッグが可能となる。

次に、ユーザが第１８図の手順画面において状況表示を選択した場合の処理について説明する。ユーザが第１８図の手順画面において状況表示を選択すると、状況表示インタフェースプログラム２２４が起動される。この状況表示インタフェースプログラム２２４は、第２４図に示す状況表示選択画面の表示制御データをＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は受取った状況表示選択画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取った状況表示選択画面の表示制御データに基づいて画面を表示する。

この画面でユーザが論理ＣＯＲＥ作成を選択すると、今度は状況表示インタフェースプログラム２２４が第２５図に示す論理ＣＯＲＥ生成状況画面の論理ＣＯＲＥの名称表示がない表示制御データを生成してＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は受取った論理ＣＯＲＥ生成状況画面の表示制御データをＷｅ

ｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取った論理ＣＯＲＥ生成状況画面の表示制御データに基づいて画面を表示する。

この画面でユーザがＰｒｏｊｅｃｔ名を入力し表示ボタンを選択すると、Ｐｒｏｊｅｃｔ名をＷｅｂクライアント１００がＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０からＰｒｏｊｅｃｔ名を受取った状況表示インタフェースプログラム２２４は、指定のＰｒｏｊｅｃｔ名のディレクトリから第１４図のディレクトリＡＳＩＣ５０にある論理ＣＯＲＥのディレクトリ名称を抽出し、論理ＣＯＲＥ生成状況画面の表示制御データを更新してＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は受取った論理ＣＯＲＥ生成状況画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取った論理ＣＯＲＥ生成状況画面の表示制御データに基づいて画面を表示する。この画面には先に生成されたＡＳＩＣの論理ＣＯＲＥのディレクトリ名称が表示される。

ここで、ＡＳＩＣの論理ＣＯＲＥ生成状況表示画面での処理手順について説明する。第２６図は、ＡＳＩＣの論理ＣＯＲＥ生成状況表示画面での処理手順を示すフローチャートである。同図に示すように、ユーザが、表示された論理ＣＯＲＥのどれかを選びダウンロードを選択すると（ステップＳ１５０１）、対象の論理ＣＯＲＥをＷｅｂクライアント１００にダウンロードすることができる（ステップＳ１５０２）。このダウンロードした論理ＣＯＲＥは、ＡＳＩＣのチップレベルのネットリストとして論理検証に使用することができ、先に述べたようにインスタンスを構成する機能に絞ったデバッグが可能となる。

一方、ユーザが論理ＣＯＲＥ名称を指定してＦＰＧＡ化ボタンを選択すると（ステップＳ１５０３）、Ｗｅｂクライアント１００は、Ｐｒｏｊｅｃｔ名と論理ＣＯＲＥ名称とＦＰＧＡ化が選択されたこととをＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０は受取ったデータを状況表示インタフェースプログラム２２４へ送る。状況表示インタフェースプログラム２２４は、第１４図のディレクトリＡＳＩＣ５０内にある、対象とする論理ＣＯＲＥ名称のディレクトリ下のＩＯディレクトリ内にあるテーブルデータのファイル名を抽出し、その結果、第２７図

に示すようなＦＰＧＡ論理ＣＯＲＥ生成インタフェース画面の表示制御データを作成しＷｅｂサーバ２２０へ送る。

Ｗｅｂサーバ２２０は受取ったＦＰＧＡ論理ＣＯＲＥ生成インタフェース画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取ったＦＰＧＡ論理ＣＯＲＥ生成インタフェース画面の表示制御データに基づいて画面を表示する（ステップＳ１５０４）。この画面では、指定されたＰｒｏｊｅｃｔ名と、論理ＣＯＲＥ名と、論理ＣＯＲＥを構成するインスタンス名のリストが左側のリストボックスにリストされる。また、この画面では、対象の論理ＣＯＲＥを変えることも可能であり、ユーザがＰｒｏｊｅｃｔ名と論理ＣＯＲＥ名を指定すると第２５図で指定した同様の処理によって画面が更新される。

さらに、この画面は、対象のＡＳＩＣの論理ＣＯＲＥを構成するインスタンスのテーブルデータからＦＰＧＡの論理ＣＯＲＥを生成するインタフェースであって、ユーザは、左のリストボックスからインスタンスを指定し追加ボタンを選択することで右のリストボックスにインスタンスを追加する。この右のリストボックスにリストされたインスタンスが１つのＦＰＧＡとなる。ユーザがインスタンスを選択し、ＦＰＧＡ名と管理のための追い番号である１以上の一意の整数であるＦＰＧＡ番号を設定して実行ボタンを選択すると、Ｗｅｂクライアント１００が、対象のＰｒｏｊｅｃｔ名と、ＡＳＩＣの論理ＣＯＲＥ名と、ＦＰＧＡ名と、ＦＰＧＡとするインスタンス名のリストと、ＦＰＧＡの番号とをＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は、受取ったデータを状況表示インタフェースプログラム２２４へ送る。

そして、状況表示インタフェースプログラム２２４は、受取ったインスタンス名のリストに従って第１４図のディレクトリＦＰＧＡ４９に指定されたＦＰＧＡの番号でディレクトリを作成し、第１４図のディレクトリＡＳＩＣ５０にある、ＣＯＲＥ名称のディレクトリ下のＩＯディレクトリから名称の一致するテーブルデータを前記番号のディレクトリへ複写し、テーブルデータ名のリストを引数としてＦＰＧＡ論理ＣＯＲＥ生成プログラム２５２を実行する。

ここで、ＦＰＧＡ論理ＣＯＲＥ生成プログラム２５２の処理手順について説明する。第２８図及び第２９図は、ＦＰＧＡ論理ＣＯＲＥ生成プログラム２５２の処理手順を示すフローチャートである。第２８図は第２１図に示した処理と同じであり、また、第２９図は第２２図に示した処理とステップＳ１７１７及びステップＳ１７２０を除いて同じであるので、この２つのステップについて説明する。

第２２図のステップＳ１０１７では、対象のインスタンスの入力であるポートに定義してある接続相手のインスタンス名とその出力ポート名が、メモリに格納している情報にない場合、エラーとしてログファイルに出力しているが、第２９図のステップＳ１７１７では、元々ＡＳＩＣの論理ＣＯＲＥとしてインスタンスのポート間接続は保証できているので、このステップＳ１７１７ではエラーではなくパッケージの入力端子情報としてメモリに追加格納する。

また、第２２図のステップＳ１０２０では、最終的に残った未接続のポート情報をログファイルに出力している。一方、ステップＳ１７２０では、第１４図のディレクトリＡＳＩＣ５０下の”ＣＯＲＥ＋プロセスＩＤ＋日付”で作成している対象とするＡＳＩＣの論理ＣＯＲＥのディレクトリにあるログファイルとステップＳ１７２０で出力したログファイルとを比較し、ステップＳ１０２０で出力したログファイルにない出力ポートをパッケージの出力端子情報としてメモリに追加格納する。後は、ステップＳ１０２０の処理と同様にして第１４図のディレクトリＦＰＧＡ５１の下の指定された番号のディレクトリ内に指定されたＦＰＧＡ名称で論理ＣＯＲＥをファイル出力する。

そして、状況表示インタフェースプログラム２２４は、ＦＰＧＡの論理ＣＯＲＥファイルを読込み、パッケージの入出力となる端子数をカウントし、第３０図に示す互換パッケージのテーブルの第２カラムにあるＩＯ数に占める割合を全て計算し、割合が最大のパッケージとその互換パッケージデータを抽出し、第２７図に示したＦＰＧＡ論理ＣＯＲＥ生成インタフェース画面のパッケージ名、ＩＯ使用率の画面表示制御データを更新し、Ｗｅｂサーバ２２０へ送る。Ｗｅｂサー

バ２２０は受取ったＦＰＧＡ論理ＣＯＲＥ生成インタフェース画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取ったＦＰＧＡ論理ＣＯＲＥ生成インタフェース画面の表示制御データに基づいて入力画面を更新する。

ユーザは、この結果に満足できなければ再度ＦＰＧＡ論理ＣＯＲＥ生成をやり直す。ユーザが決定を選択すると、Ｗｅｂクライアント１００は決定が選択されたことと、ＦＰＧＡ番号とパッケージ名とＩＯ使用率のデータとをＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０からこれらのデータを受取った状況表示インタフェースプログラム２２４は、第１４図のディレクトリＦＰＧＡ５５に指定された番号のディレクトリを作成し、例えばplistの名称で受取ったデータをファイル出力する。

次に、論理合成インタフェースプログラム２２２の処理手順について説明する。第３１図は、論理合成インタフェースプログラム２２２の処理手順を示すフローチャートである。同図に示すように、この論理合成インタフェースプログラム２２２は、第３２図に示すような論理合成インタフェース画面の表示制御データをＷｅｂサーバ２２０へ送り、Ｗｅｂサーバ２２０は受取った論理合成インタフェース画面の表示制御データをＷｅｂクライアント１００へ送り、Ｗｅｂクライアント１００は受取った論理合成インタフェース画面の表示制御データに基づいて画面を表示する（ステップＳ２００１）。

ユーザがＰｒｏｊｅｃｔ名、第２５図で表示されている対象の論理ＣＯＲＥ名、論理合成する対象の機能ブロック名、組込むＦＰＧＡの番号を指定し、かつ面積優先か速度優先かの指定と、Ｄｅｂｕｇ（ＦＰＧＡの論理合成を実行しない）かＦｉｘ（ＦＰＧＡの論理合成を実行する）かを指定し（ステップＳ２００２）、実行ボタンを選択すると（ステップＳ２００３）、Ｗｅｂクライアント１００はデータ選択画面を表示する。

ユーザがこの画面に従って論理合成するＲＴＬソースを選択すると、これらのデータをＷｅｂクライアント１００はＷｅｂサーバ２２０に送り、Ｗｅｂサーバ

２２０からデータを受取った論理合成インタフェースプログラム２２２は、ＲＴＬソースを第１４図のディレクトリＡＳＩＣ５２に指定の機能ブロック名でディレクトリを作成し、ＲＴＬを格納する（ステップＳ２００４）。そして、ＲＴＬソースを１ファイルずつ読込み、ＲＴＬソース内の変更履歴、版数などを記載するヘッダー部分にユーザが定義している動作周波数データを抽出し、抽出した値に対して２０％増した周波数値とユーザに指定された速度優先または面積優先の論理合成条件を論理合成の制約として論理合成対象の機能ブロック名を指定してＡＳＩＣの論理合成ツールのＡＳＩＣ論理合成プログラム２５３を実行する（ステップＳ２００５）。

ＡＳＩＣの論理合成が終了すると、論理合成インタフェースプログラム２２２は、第１４図のディレクトリＡＳＩＣ５４下にある指定の機能ブロックのディレクトリにＡＳＩＣ論理合成プログラム２５３が出力した結果のレポートファイルと、ユーザ管理サーバ２４０にＰｒｏｊｅｃｔ名で問い合わせて得たユーザのメールアドレスとブロック名とをメールサーバ２６０に送り、メールサーバ２６０からユーザにメールを送信する（ステップＳ２００６）。

そして、論理合成インタフェースプログラム２２２は、この処理過程を第１４図のディレクトリＡＳＩＣ５２下に指定の機能ブロック名で作成したディレクトリに存在する全ＲＴＬソースに対して実行する。全ＲＴＬソースの論理合成が終了すると（ステップＳ２００７の肯定）、論理合成インタフェースプログラム２２２は、速度優先指定の場合は（ステップＳ２００８の肯定）、全ＲＴＬの論理合成結果のレポートを検索し、ＲＴＬソースに定義された動作周波数を満足しているか否かを判断する（ステップＳ２００９～ステップＳ２０１０）。面積重視の場合は何もしない。

次に、Ｆｉｘ指定を調べ（ステップＳ２０１１）、Ｆｉｘ指定されていれば、論理合成インタフェースプログラム２２２は、指定されたＦＰＧＡ番号で第１４図のディレクトリＦＰＧＡ５５下にディレクトリを作成し、この中に第１４図のディレクトリＡＳＩＣ５２下にある指定された機能ブロックディレクトリ内のＲ

ＴＬを複写して格納し、指定されたＰｒｏｊｅｃｔ名とＦＰＧＡ番号とを引数としてＦＰＧＡ論理合成インタフェースプログラムを実行する（ステップＳ２０１２）。

そして、論理合成インタフェースプログラム２２２は、論理合成ツールを起動し、第１４図のディレクトリＡＳＩＣ５０にある指定の論理ＣＯＲＥのディレクトリからＡＳＩＣの論理ＣＯＲＥを論理合成ツールにコマンドを投入して読込ませ、ユーザからアップロードされ第１４図のディレクトリＡＳＩＣ５４に存在するＡＳＩＣのパッケージ端子割付けとＡＳＩＣと接続される外部デバイスと電気的にインタフェースするバッファとを定義したファイルを読込み、定義に従って論理合成ツールにコマンドを投入することでＡＳＩＣのバッファとスキャンなどテスト回路をＡＳＩＣの論理ＣＯＲＥの端子とパッケージの端子と接続されるチップのパッド間に挿入し接続させ、そして、論理合成済みである第１４図のディレクトリＡＳＩＣ５２下の全ディレクトリにある機能ブロックのデータを論理合成ツールにコマンドを投入して読込んで論理ＣＯＲＥに填め込ませ、レイアウト設計用にＡＳＩＣのネットリストを生成させ、第１４図のディレクトリＡＳＩＣ５４にＰｒｏｊｅｃｔ名でファイル出力させる（ステップＳ２０１３）。

次に、ＦＰＧＡ論理合成インタフェースプログラムの処理手順について説明する。第３３図は、ＦＰＧＡ論理合成インタフェースプログラムの処理手順を示すフローチャートである。同図に示すように、このＦＰＧＡ論理合成インタフェースプログラムは、ＡＳＩＣの論理合成インタフェースプログラムから指定されたＦＰＧＡ番号で第１４図のディレクトリＦＰＧＡ５５にディレクトリを作成し、第１４図のディレクトリＦＰＧＡ５３下にある指定された番号のディレクトリ内のＲＴＬソースを１ファイルずつ読込み、ＲＴＬソース内の変更履歴、版数などを記載するヘッダー部分にユーザが定義している動作周波数データを抽出し、抽出した値に対して２０％増した周波数値を論理合成の制約として論理合成対象の機能ブロック名を指定してＦＰＧＡの論理合成ツールのＦＰＧＡ論理合成プログラム２５４を実行する（ステップＳ２２０１）。

ＦＰＧＡの論理合成が終了すると、ＦＰＧＡ論理合成プログラム２５４は、第１４図のディレクトリＦＰＧＡ５５にある対象とする番号のディレクトリに論理合成結果を機能ブロックの名称で出力する。こうすることでユーザは、後述する第３４図の論理合成状況表示でゲート使用率を確認しつつ設計作業を進めることができる。そして、ＦＰＧＡ論理合成インタフェースプログラムは、論理合成の終了メッセージと、ＦＰＧＡの番号と、ブロック名と、Ｐｒｏｊｅｃｔ名でユーザ管理サーバ２４０に問い合わせ得たユーザのメールアドレスとをメールサーバ２６０に送り、メールサーバ２６０からユーザにメールを送信する（ステップＳ２２０２）。

次に、ＦＰＧＡ論理合成インタフェースプログラムは、指定された番号のＦＰＧＡを構成する機能ブロックの論理合成がすべて終了しているか確認する（ステップＳ２２０３）。この確認は、第１４図のディレクトリＦＰＧＡ５１の対象とする番号のディレクトリにある機能ブロックの名称を抽出し、前記論理合成が終了した機能ブロック名と比較することで行なう。差分がなければ、次にＦＰＧＡ論理合成インタフェースプログラムは、今度は、第１４図のディレクトリＦＰＧＡ５１にある対象の番号のディレクトリにあるＦＰＧＡ名である論理ＣＯＲＥのファイル名を抽出し、この名称で第１４図のディレクトリＦＰＧＡ５５にＦＰＧＡのレイアウトである、フィッティングに使用するＦＰＧＡの論理ＣＯＲＥのポート名と同一であるパッケージの端子名と端子番号の対応、動作周波数、などを定義したフィッティング制御ファイルが、ユーザからアップロードされ存在しているかチェックする（ステップＳ２２０６）。

存在していれば、ＦＰＧＡ論理合成インタフェースプログラムは、ＦＰＧＡを構成する全機能ブロックと第１４図のディレクトリＦＰＧＡ５１下の、対象の番号のディレクトリにあるＦＰＧＡの論理ＣＯＲＥとを指定し、ＦＰＧＡ論理合成プログラム２５４を実行する。そしてＦＰＧＡの論理合成が終了すると、ＦＰＧＡ論理合成インタフェースプログラムは、フィッティングプログラム２５５を起動し、論理合成結果のネットリストをフィッティングプログラム２５５に投入し

てレイアウトを行ない、ＦＰＧＡのレイアウト結果であり回路情報を記録したＲＯＭデータを生成し、第１４図のディレクトリＲＯＭ４６下の対象とする番号のディレクトリにＦＰＧＡ名で出力させる（ステップＳ２２０７）。そして、ＲＯＭデータ生成の終了メッセージと、ＦＰＧＡの番号と、Ｐｒｏｊｅｃｔ名と、ユーザ管理サーバ２４０に問い合わせて得たユーザのメールアドレスとをメールサーバ２６０へ送り、メールを送信する（ステップＳ２２０８）。

差分があれば、後述するフィッティングデータの投入と実行画面で実行ボタンが選択されていないか判断し（ステップＳ２２０４）、選択されていなければ終了するが、選択されていれば論理合成が未の機能ブロック全てに対し、第１４図のディレクトリＦＰＧＡ５１下にある対象の番号のディレクトリにある、ＦＰＧＡの論理ＣＯＲＥ内の各機能ブロック内部で、入出力ポート全てにフリップ・フロップを接続し、入力側の全フリップ・フロップの出力ポートを適当なゲート、例えば２入力ＮＡＮＤの入力ポートに接続し、全ＮＡＮＤゲートの出力を新たな２入力のＮＡＮＤゲートの入力ポートに接続し、前述のように２入力ＮＡＮＤの多段接続を繰り返し、最終段の２入力ＮＡＮＤの出力を出力側の全フリップ・フロップの入力ポートに接続するＨＤＬ記述を挿入して論理合成可能な状態にする（ステップＳ２２０５）。

後は、差分がない場合と同一の処理となる。開発着手時点の検証戦略にも依るが、回路アーキテクチャ検討にて機能的に上りと下りに大きく分割できる機能ブロック構成となっている場合、上述のフリップ･フロップなどを用いたダミー回路を挿入することによって、上りの機能の設計が完了していれば、下りの機能の設計が未完であっても検証が可能であり、検証の効率化に貢献するところが大きい。

次に、第３４図の論理合成状況表示画面について説明する。この画面は、ユーザがＷｅｂクライアント１００で第２４図の状況表示画面にある論理合成を選択すると、Ｗｅｂクライアント１００は論理合成を選択されたことをＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０からデータを受取った状況表示インタフェー

スプログラム２２４は、ログインしたユーザの名称でユーザ管理サーバ２４０に問い合わせて得たＰｒｏｊｅｃｔ名全てに対し、第１４図のディレクトリＦＰＧＡ５５下の、全番号のディレクトリにある論理合成結果からゲート規模を抽出する。

そして、先に説明したＦＰＧＡ論理ＣＯＲＥ生成で作成された同一ディレクトリにあるplistからパッケージ情報を抽出し、このパッケージ情報を基に第３０図のテーブル内でゲート規模の割合が７５％以下になるパッケージを選び出し、第３４図の論理合成状況表示画面の表示制御データを生成し、Ｗｅｂサーバ２２０に送る。Ｗｅｂサーバ２２０は、論理合成状況表示画面の表示制御データをＷｅｂクライアント１００に送り、Ｗｅｂクライアント１００は、受取った論理合成状況表示画面の表示制御データに基づいて画面を表示する。

ここで、ゲート規模の割合７５％について簡単に説明しておく。通常ＦＰＧＡの論理合成ツールは、論理合成の結果のゲート規模を基に論理合成ツールがもつデータベースから適当なパッケージとゲート使用率を示唆してくれる。しかしながら、ＦＰＧＡのフィッティングにおいては、セルの配置過程において配線の効率を上げるため、ＦＰＧＡで論理を実現する単位ブロックを配線に使用する。従って、論理合成結果よりゲート規模が大きくなることを考慮し、フィッティング後ゲート使用率が１００％を超えないように７５％に設定しているものであり、ＦＰＧＡで異なることもある。このためにも先に説明したＦＰＧＡの論理合成を機能ブロック毎に実行させることは有効である。

次に、フィッティングデータの投入と実行画面について説明する。第３５図は、フィッティングデータの投入と実行画面の一例を示す図である。ユーザがＷｅｂクライアント１００で第１８図に示した手順画面にあるフィッティングを選択すると、Ｗｅｂクライアント１００はフィッティングが選択されたことをＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０からデータを受取ったフィッティングインタフェースプログラム２２３は、ログインしたユーザの名称でユーザ管理サーバ２４０に問い合わせて得たＰｒｏｊｅｃｔ名全てに対し、第１４図のディレ

クトリＦＰＧＡ５５下の、全ディレクトリの番号名と、それぞれのディレクトリにある論理合成結果であるネットリストのファイル名即ちＦＰＧＡ名を抽出し、第３５図に示すフィッティングデータの投入と実行画面の表示制御データを生成し、Ｗｅｂサーバ２２０に送る。

Ｗｅｂサーバ２２０はフィッティングデータの投入と実行画面の表示制御データをＷｅｂクライアント１００に送り、Ｗｅｂクライアント１００は、受取ったフィッティングデータの投入と実行画面の表示制御データに基づいて画面を表示する。この画面の条件データ投入を選択すると、Ｗｅｂクライアント１００によりデータ選択画面が表示される。

この画面に従い、ユーザが先に述べたフィッティング制御ファイルを選択すると、Ｗｅｂクライアント１００は、選択したＦＰＧＡの番号とフィッティング制御ファイルをＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０からデータを受取ったフィッティングインタフェースプログラム２２３は、第１４図のディレクトリＦＰＧＡ５５下の、対象の番号のディレクトリにフィッティング制御ファイルを格納する。

また、ユーザが実行ボタンを選択すると、Ｗｅｂクライアント１００は、選択したＦＰＧＡの番号と、Ｐｒｏｊｅｃｔ名と、実行が選択されたことをＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０からデータを受取ったフィッティングインタフェースプログラム２２３は、Ｐｒｏｊｅｃｔ名とＦＰＧＡ番号とを引数としてＦＰＧＡ論理合成インタフェースプログラムを実行する。ＦＰＧＡ論理合成インタフェースプログラムの処理は先に説明した通りである。

次に、ＲＯＭデータ生成状況画面について説明する。第３６図は、ＲＯＭデータ生成状況画面の一例を示す図である。ユーザがＷｅｂクライアント１００で第２４図に示した状況表示選択画面にあるＲＯＭデータを選択すると、Ｗｅｂクライアント１００はＲＯＭデータを選択されたことをＷｅｂサーバ２２０に送る。

Ｗｅｂサーバ２２０からデータを受取った状況表示インタフェースプログラム２２４は、ログインしたユーザの名称でユーザ管理サーバ２４０に問い合わせて

得たＰｒｏｊｅｃｔ名全てに対し、第１４図のディレクトリＲＯＭ４６下の、全ディレクトリの番号名と、それぞれのディレクトリにあるフィッティング結果であるＲＯＭデータ名即ちＦＰＧＡ名を抽出し、第３６図に示したＲＯＭデータ生成状況画面の表示制御データを生成し、Ｗｅｂサーバ２２０に送る。

Ｗｅｂサーバ２２０は、ＲＯＭデータ生成状況画面の表示制御データをＷｅｂクライアント１００に送り、Ｗｅｂクライアント１００は、受取ったＲＯＭデータ生成状況画面の表示制御データに基づいて画面を表示する。ユーザは、この画面で対象とするＦＰＧＡの日付部分を選択すると、ＲＯＭデータをダウンロードすることができる。

次に、監視サーバ２９０の処理について説明する。先ず、監視サーバ２９０が監視するものは、ＲＴＬの変更有無と、論理合成、フィッティング、レイアウト設計ツールの処理に要した時間と、設計者が定義する機能ブロックの変更規模を基にインプリメント設計に使用するツールの処理時間以外にかかる、変更反映に必要な手作業の時間である。

インプリメント設計者が管理する第１４図のディレクトリＬＡＹＯＵＴ４７には、Ｐｒｏｊｅｃｔ毎にＰｒｏｊｅｃｔ名のディレクトリがあり、その中で作業を進める。各Ｐｒｏｊｅｃｔ名のディレクトリにはレイアウト設計データが格納されるＬＡＹディレクトリとＲＴＬソースを格納するＲＴＬディレクトリが存在し、ＲＴＬディレクトリ以下は、第１４図のディレクトリＡＳＩＣ５２以下と同一のディレクトリ構成となっている。

監視サーバ２９０は、ＬＡＹとＲＴＬディレクトリ内を監視する。時間を集計した結果を反映するスケジュールは、Ｐｒｏｊｅｃｔ名のディレクトリにＳｃｈｅｄｕｌｅの名称で格納されており、初期状態では、契約に基づいて、１ｓｔＲＴＬの予定日とＳｉｇｎ Ｏｆｆの予定日が設定されている。１ｓｔＲＴＬとは、レイアウト設計が着手できる、その時点で大きな変更の見込みがなく、機能検証が８０％程度終了しているＲＴＬであると定義する。Ｓｉｇｎ Ｏｆｆとは、ＡＳＩＣのレイアウト設計完了後、ＡＳＩＣ製造データをデバイスベンダに渡す

日である。

第３７図は、予定と実績画面の一例を示す図である。同図は、Ｓｃｈｅｄｕｌｅファイルを画面表示制御データに変換し、Ｗｅｂクライアント１００によって表示される。すなわち、ユーザが第２４図の状況表示画面で実績と予定を選択すると、Ｗｅｂクライアント１００は、実績と予定が選択されたことをＷｅｂサーバ２２０経由で状況表示インタフェースプログラム２２４に送る。実績と予定が選択されたことを受け取った状況表示インタフェースプログラム２２４は、Ｓｃｈｅｄｕｌｅファイルを画面表示制御データに変換し、Ｗｅｂサーバ２２０経由でＷｅｂクライアント１００に送る。画面表示制御データを受け取ったＷｅｂクライアント１００は、画面表示制御データに基づいて画面表示する。

また、第３８図は、レイアウト設計者が設定する作業時間の設定ファイルの一例を示す図である。このファイルは、Ｓｃｈｅｄｕｌｅファイルと同一ディレクトリにＭａｎｕａｌの名称で格納される。

監視サーバ２９０は、監視時間になると、ユーザ管理サーバ２４０からユーザ名とユーザが所属するＰｒｏｊｅｃｔ名とメールアドレスを獲得する。そして、Ｓｃｈｅｄｕｌｅファイルを読みこみ１ｓｔＲＴＬの予定日を抽出する。この予定日が監視の実行年月日より未来であれば何もせず終了する。予定日が監視の実行年月日より以前であれば、以降の処理を実行する。

まず、インプリメント設計者が管理するＬＡＹＯＵＴディレクトリ下の、Ｐｒｏｊｅｃｔ名のディレクトリにあるＲＴＬディレクトリにＲＴＬソースがなければ処理を中断する。ＲＴＬソースがあれば、第１４図のディレクトリＡＳＩＣ５２下の、各ブロックのディレクトリにあるＲＴＬソース間で比較する。

差分があれば、差分のあるブロック名とＰｒｏｊｅｃｔ名とインプリメント設計者のメールアドレスをメールサーバ２６０に送り、メールサーバ２６０からインプリメント設計者に送信する。インプリメント設計者は、このメールによって変更の反映にかかる時間の見積り作業を行なう。見積りの結果、インプリメント設計者は、必要があればＳｃｈｅｄｕｌｅファイルの時間データを更新または追

加、削除する。

次に、監視サーバ２９０は、第１４図のディレクトリＡＳＩＣ５０下の、作成時間が一番新しいＡＳＩＣの論理ＣＯＲＥからＡＳＩＣを構成するブロック名を抽出し、メモリに格納する。そして、第１４図のディレクトリＡＳＩＣ５４下の、各ブロックのディレクトリにある論理合成結果のレポートファイルから論理合成処理時間を抽出する。先にメモリに格納したブロック名に対するレポートファイルが存在しない、つまり未だ論理合成されていない場合は、存在するレポートファイルから抽出した時間の平均時間を適用し、合計時間を算出しメモリに格納する。

そして今度は、インプリメント設計者が管理する第１４図のディレクトリＬＡＹＯＵＴ４７下の、Ｐｒｏｊｅｃｔ名のディレクトリ内にあるレイアウト設計結果のうち、レイアウト設計工程の最終工程であるタイミング検証結果のファイルが存在すれば、レイアウト設計の全行程で使用するツールが出力した処理結果ファイルから処理時間を抽出し、先にメモリに格納した時間と合せて合計時間を算出する。タイミング検証結果のファイルが存在しない場合は、契約時点にレイアウト設計者が仮に見積り予め監視サーバに設定した時間を適用する。

次に、監視サーバ２９０は、この合計時間と先に説明したＭａｎｕａｌに設定されている時間の合計をとり、１日を１２時間として日数を算出し、Ｓｃｈｅｄｕｌｅファイルを必要であれば更新する。つまり、最初の状態では、Ｓｃｈｅｄｕｌｅファイルには１ｓｔＲＴＬの予定日しかなく、この予定日に対して合計時間を加算し、第３７図に示したように反映開始日としてインプリメント設計への変更取り込みの予定日を追加する。

次の監視では、反映開始日が監視の実行年月日より未来であれば、反映開始日の一つ前である１ｓｔＲＴＬ予定日から反映開始日として設定した予定日までの日数を比較対照とし、算出したインプリメント設計にかかる日数より長ければ反映開始日は更新しない。反映開始日が監視の実行年月日以前の時、監視サーバ２９０は、ユーザとインプリメント設計者のメールアドレスと反映開始の機会であ

るメッセージとをメールサーバ２６０へ送り、メールサーバ２６０からユーザとインプリメント設計者にメールを送信する。この時監視サーバ２９０は、一番新しい反映開始日が監視実行の年月日以前である時、反映開始日の実績日に年月日がなければＳｃｈｅｄｕｌｅファイルの更新はしない。

反映開始の機会であるメッセージのメールに対してユーザ及びインプリメント設計者双方から承認または拒絶のメールが監視サーバ２９０に届いた時、双方共承認であれば直ちにＳｃｈｅｄｕｌｅファイルの反映開始日の実績日にメールを受けた年月日を設定し、次回の監視では、一番新しい反映予定日が監視実施の年月日以前であり、反映予定日の実績日に年月日があれば、実績日を基にインプリメント設計の日数を算出し、新たに反映開始日を追加する。この処理を続けて行くと反映開始日がＳｉｇｎ　Ｏｆｆ日に近づく。

反映開始日の予定日を算出し、Ｓｉｇｎ　Ｏｆｆ日を超えた場合、監視サーバ２９０は、反映開始の予定日の設定が不可能な年月日にきているメッセージとユーザとインプリメント設計者のメールアドレスをメールサーバ２６０へ送り、メールサーバ２６０から双方へメールを送信する。

反映開始の機会であるメッセージのメールに対してユーザが拒絶している場合、ユーザに対しＷｅｂクライアント１００から予定日を設定する旨のメッセージとユーザとインプリメント設計者のメールアドレスをメールサーバ２６０へ送り、メールサーバ２６０から双方へメールを送信する。

次に、予定と実績画面ついて説明する。第３７図は、予定と実績画面の一例を示す図である。同図の論理Ｆｉｘボタンと変更ボタンは、監視サーバ２９０の監視の実行年月日以前であり、一番新しい反映開始日のよこに配置される。ユーザがこのボタンの下にある年月日の設定欄に年月日を指定し変更ボタンを選択すると、Ｗｅｂクライアント１００は、設定された年月日と変更が選択されたことをＷｅｂサーバ２２０に送り、Ｗｅｂサーバ２２０からデータを受取った状況表示インタフェースプログラム２２４は、Ｓｃｈｅｄｕｌｅファイル内の一番新しい反映開始日の予定日を指定された年月日に変更し、変更された旨のメッセージと

ユーザとインプリメント設計者のメールアドレスをメールサーバ２６０へ送り、メールサーバ２６０から双方へメールを送信する。

第３７図内の論理Ｆｉｘボタンが年月日の指定とともに選択された場合は、状況表示インタフェースプログラム２２４は、変更ボタンを選択された場合と同一の処理を実行後、監視サーバ２９０の監視を停止する処理を行なう。こうする事でユーザとインプリメント設計者双方の短期間の目標が明確となることと、インプリメント設計の試行回数が多いほど変更開始日の予定日の確度があがり、これに対してユーザ及びインプリメント設計者双方共に進め方を考える契機を与える事が可能となり、結果として効率の向上に大きく貢献する。

上述してきたように、本実施の形態２では、Ｗｅｂクライアント１００からのユーザの要求に基づいてＡＳＩＣ論理合成プログラム２５３がＡＳＩＣの論理合成を行ない、ＦＰＧＡ論理合成プログラム２５４がＦＰＧＡの論理合成を行ない、論理合成インタフェースプログラム２２２がＷｅｂクライアント１００にＡＳＩＣ及びＦＰＧＡの論理合成の結果を表示し、メールサーバ２６０がＡＳＩＣ及びＦＰＧＡの論理合成の開始と結果をユーザに電子メールで通知することとしたので、ユーザは、論理合成の専任者を設置する必要がなく、いつでも論理合成が実行でき、専任者が実行したが如く論理合成品質を均一に保つことができるとともに、電子メールで論理合成の開始と結果の通知を受けることができ、論理合成の進捗を定期的にコンピュータで確認する必要がなくなる。

また、本実施の形態２では、Ｗｅｂクライアント１００からのユーザの要求に基づいて、ＡＳＩＣ論理ＣＯＲＥ生成プログラム２５１がＡＳＩＣを構成する機能ブロックからユーザが指定する複数の機能ブロックのポート間接続情報のみから成るネットリストを生成し、ＦＰＧＡ論理ＣＯＲＥ生成プログラム２５２が生成したネットリストに論理合成済み対象機能ブロックのデータを埋め込んでプログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを生成し、論理ＣＯＲＥ生成インタフェースプログラム２２１が生成したＲＯＭデータの生成結果をコンピュータに表示するとともにメールサーバ２６０がユーザに電子メールで

通知する構成としたので、ユーザは、プログラマブル・ロジックデバイス専用の開発環境が必要なく、プログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを作成する負荷と、時間と、かかるコストとを削減することができる。

また、本実施の形態２では、ユーザが指定したＡＳＩＣを構成する機能ブロックの設計が未完了で回路データが存在しない場合に、ＦＰＧＡ論理ＣＯＲＥ生成プログラム２５２が当該機能ブロックの入力端子及び出力端子に仮のフリップ・フロップなどを用いた回路を挿入したネットリストを生成することとしたので、プログラマブル・ロジックデバイスのプロトタイピングによる検証において、検証対象ではない機能ブロックの設計が完了していなくとも、プロトタイピングによる検証を進めることができ、検証の効率化を図ることができる。

また、本実施の形態２では、監視サーバ２９０が、ユーザが持つ最新の回路データとインプリメント設計者がインプリメント設計に取り込んでいる回路データ間の変更の規模を監視し、監視結果及びレイアウト設計にかかる時間に基づいて計画された日時に到達すると、メールサーバ２６０が変更をＡＳＩＣのインプリメント設計へ反映するタイミングにあることをユーザ及びＡＳＩＣのインプリメント設計者に電子メールで通知し、この通知に応答してユーザが反映の日付を変更することで停止を要求することとしたので、発生した変更を効率よくＡＳＩＣのレイアウト設計に反映することができるとともに、変更を反映するタイミングを設定することで、ユーザはいつまで変更することが可能かを判断することができ、早い段階でスケジュールの見直しができる。

以上説明したように、本発明によれば以下の効果が得られる。

請求の範囲第１項記載の発明によれば、請求の範囲第１項の発明のコア（論理コア）を生成する手法をコンピュータに実行させるプログラムを記憶したプログラム記憶媒体によれば、集積回路開発において、ＲＴＬ設計の入力となるブロックのポート仕様の品質を事前に確保できる効果があり、更に機能ブロックが多く存在し、設計リソースも多い大規模集積回路の開発においても、事前にブロック間の接続が確認できていることにより、必ずチップが組み上がることを保証する

ことができる。

請求の範囲第２項記載の発明によれば、ＡＳＩＣのコア（論理コア）からプログラマブル・ロジックデバイスのコア（論理コア）としてブロック間接続を保持したまま切り出すため、本コア（論理コア）を用いたプログラマブル・ロジックデバイスの検証を実施すれば、ＡＳＩＣにおいては、少なくとも同一の構成での検証の重複を回避することができる。よって、ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発を効率よく進めることができる。

また、Ｉ／Ｏバッファを挿入する際、チップの端子情報から仮のコア（論理コア）を生成し、ブロックから生成したコア（論理コア）と入れ替える手段によって、ブロックに定義されているチップのポート情報とチップの端子情報をクロスチェックでき、ブロックのポート仕様とチップ端子仕様の両者の品質を確保することができる。

請求の範囲第３項記載の発明によれば、任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストを生成することができる。

請求項４記載の発明によれば、任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストから、チップネットリストを生成することができる。

請求の範囲第５項記載の発明によれば、プログラマブル・ロジックデバイスのコア（論理コア）を生成する手法を論理合成ツールに実行させるプログラムを記憶した記憶媒体によれば、ポートの機能を判断することが難しいネット名がポートの名称となる場合の論理検証等デバッグの効率の低下を防ぐことができる。

上述のようにして、設計ドキュメントのデータからコア（論理コア）を生成し、該コア（論理コア）から階層構造及び接続情報を保持した状態でプログラマブル・ロジックデバイス用に新たにコア（論理コア）を生成することで、回路アーキテクチャを共有することができ、回路データを填め込み、機能検証したインスタンス内のデバイス・テクノロジに依存しない回路データ及びインスタンス間のネットは、ＡＳＩＣ化した場合に再検証を回避することが可能となり、ＡＳＩＣ

とプログラマブル・ロジックデバイス間の違いによる再設計も回避することが可能となる。

このように、本発明によれば、アーキテクチャの共有化を実現し、再設計と再検証を極力回避することができる集積回路の開発方法及び集積回路の開発方法を記憶したプログラマブル記憶媒体を提供することができる。

また、本発明によれば、ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成し、ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成し、グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを生成したネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成し、作成したＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を実機評価用ＲＯＭデータの生成とコンカレントに行ない、生成したＲＯＭデータを用いた実機評価結果に基づく回路データの変更をＡＳＩＣのレイアウト作成およびタイミング検証に反映することとしたので、ＡＳＩＣとプログラマブル・ロジックデバイスの効率的なコンカレント開発が可能になり、ＡＳＩＣの開発期間を短縮することができるという効果を奏する。

また、本発明によれば、ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成し、ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成し、グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを生成したネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成し、作成したＡＳ

ＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を実機評価用ＲＯＭデータの生成とコンカレントに行なうよう構成したので、ＡＳＩＣとプログラマブル・ロジックデバイスの効率的なコンカレント開発が可能になり、ＡＳＩＣの開発期間を短縮することができるという効果を奏する。

また、本発明によれば、ユーザの要求に応じてＡＳＩＣの論理合成を実行し、作成したＡＳＩＣの論理合成結果がユーザの要求する速度性能を満足したか否かを判断し、判断結果に基づいてプログラマブル・ロジックデバイスの論理合成を実行し、ＡＳＩＣ論理合成の実行結果及びプログラマブル・ロジックデバイス論理合成の実行結果をコンピュータに表示し、ＡＳＩＣ論理合成の実行開始と実行結果及びプログラマブル・ロジックデバイス論理合成の実行開始と実行結果をユーザに電子メールで通知する構成としたので、ユーザは、論理合成の専任者を設置する必要がなく、いつでも論理合成が実行でき、専任者が実行したが如く論理合成品質を均一に保つことができるとともに、電子メールで論理合成の開始と結果の通知を受けることができ、論理合成の進捗を定期的にコンピュータで確認する必要がなくなるという効果を奏する。

また、本発明によれば、ユーザの要求に応じてＡＳＩＣを構成する機能ブロックからユーザが指定する複数の機能ブロックのポート間接続情報から成るネットリストを生成し、生成したネットリストに論理合成済み対象機能ブロックのデータを填め込んでプログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを生成し、生成したＲＯＭデータの生成結果をコンピュータに表示するとともにユーザに電子メールで通知する構成としたので、ユーザは、プログラマブル・ロジックデバイス専用の開発環境が必要なく、プログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを作成する負荷と、時間と、かかるコストとを削減することができるという効果を奏する。

また、本発明によれば、ユーザが指定したＡＳＩＣを構成する機能ブロックの設計が未完了で回路データが存在しない場合に、当該機能ブロックの入力端子及び出力端子に仮のフリップ・フロップなどを用いた回路を挿入したネットリスト

を生成することとしたので、プログラマブル・ロジックデバイスのプロトタイピングによる検証において、検証対象ではない機能ブロックの設計が完了していなくとも、プロトタイピングによる検証を進めることができ、検証の効率化を図ることができるという効果を奏する。

また、本発明によれば、ユーザが持つ最新の回路データとインプリメント設計者がインプリメント設計に取り込んでいる回路データ間の変更の規模を監視し、監視結果及びレイアウト設計にかかる時間に基づいて計画された日時に到達すると変更をＡＳＩＣのインプリメント設計へ反映するタイミングにあることをユーザ及びＡＳＩＣのインプリメント設計者に電子メールで通知し、この通知に応答してユーザが反映の日付を変更することで停止を要求することとしたので、発生した変更を効率よくＡＳＩＣのレイアウト設計に反映することができるとともに、変更を反映するタイミングを設定することでユーザはいつまで変更することが可能かを判断することができ、早い段階でスケジュールの見直しができるという効果を奏する。

また、本発明によれば、一方では、複数の機能ブロックの全部又は一部の機能ブロックを含むＦＰＧＡの端子情報が記述された第１デザインと、第１デザインの下位層として記述されたＦＰＧＡと同様の端子情報が記述された第２デザインとから、第１デザイン、第２デザインにおける同一端子間が接続され、かつ、端子間にＦＰＧＡ対応のバッファが挿入されたＦＰＧＡ用デザイン情報を生成し、他方では、複数の機能ブロックを含むＡＳＩＣの端子情報が記述された第３デザインと、第３デザインの下位層として記述されたＡＳＩＣと同様の端子情報が記述された第４デザインとから、第３デザイン、第４デザイン間における同一端子間が接続され、かつ、端子間にＡＳＩＣ対応のバッファが挿入されたＡＳＩＣ用デザイン情報を生成し、第２デザイン、第４デザインのそれぞれを、各デザインが含む機能ブロックの接続情報に基づいて生成された回路情報で置き換える、ことでＦＰＧＡ及びＡＳＩＣのネットリストを生成することとしたので、ＡＳＩＣとプログラマブル・ロジックデバイスの効率的なコンカレント開発が可能になり

、ＡＳＩＣの開発期間を短縮することができるという効果を奏する。

産業上の利用可能性

以上のように、本発明に係る集積回路の開発方法及び集積回路の開発方法を記憶したプログラム記憶媒体、ならびにＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム、開発プログラム及び開発方法は、集積回路の開発、特にＡＳＩＣとプログラマブル・ロジックデバイスの開発に適している。

請 求 の 範 囲

1． ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックを選択しグループ化する手段により任意の規模、個数のブロックのポートとポートの接続情報からなる、論理合成ツールが読み込み可能なＨＤＬフォーマットのコア（論理コア）を生成し（ステップ１）、

チップの端子情報から論理合成ツールで仮のチップデザインを作成し、このデザインに端子を発生させ（ステップ２）、

作成したデザイン内部にステップ２と同一のデザインをセルとして発生させ（ステップ３）、

デザインとセル間の同一名称のポートを接続し（ステップ４）、

接続したポート間のネットに対し、デバイス・テクノロジに依存するＩ／Ｏバッファを挿入し（ステップ５）、

ステップ１で作成したコア（論理コア）とセルを入れ替え、トップ階層であるデザインの階層を展開し、ネットリストを生成する（ステップ６）、

ことを特徴とする集積回路の開発方法を記憶したプログラム記憶媒体。

2． ＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発を行なう場合において、

ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）と呼ぶネットリストから、接続関係のある任意のブロックを選択しグループ化する手段により、任意の規模、個数のブロックのポートとポートの接続情報からなるコア（論理コア）を生成し（ステップ１）、

チップの端子情報から論理合成ツールで仮のチップのデザインを作成し（ステップ２）、

このデザインに前記チップの端子情報にある名称で端子を発生させ（ステップ

3）、

この内部に前記と同一デザインをセルとして発生させ（ステップ４）、

前記デザインとセル間の同一名称のポートを接続し（ステップ５）、

この接続ネットに対し、チップの端子情報からデバイス・テクノロジに依存するＩ／Ｏバッファを挿入し（ステップ６）、

ステップ１で生成したコア（論理コア）を前記セルと入れ替え（ステップ７）、

このコア（論理コア）のトップ階層である、作成したデザインの階層を展開し、プログラマブル・ロジックデバイスのチップのネットリストを生成する（ステップ８）、

ことを特徴とする集積回路の開発方法。

３．　ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックを選択しグループ化する手段により、任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストを生成することを特徴とする請求の範囲第１項に記載のプログラム記憶媒体。

４．　ブロックのポートとポートの接続情報からなるＡＳＩＣのコア（論理コア）から、接続関係のある任意のブロックを選択しグループ化する手段により、任意の規模、個数のブロックのポートとポートの接続情報からなるコア（論理コア）を生成し（ステップ１）、

チップの端子情報から、論理合成ツールで仮のチップのデザインを作成し、このデザインに前記チップの端子情報にある名称で端子を発生させ（ステップ２）、

この内部に前記と同一デザインをセルとして発生させ（ステップ３）、

前記デザインとセル間の同一名称のポートを接続し（ステップ４）、

この接続ネットに対しチップの端子情報からデバイス・テクノロジに依存する

Ｉ／Ｏバッファを挿入し（ステップ５）、

ステップ１で生成したコア（論理コア）を前記セルと入れ替え（ステップ６）、

このコア（論理コア）のトップの階層であるステップ２で作成したデザインの階層を展開してチップネットリストを生成する（ステップ７）、

ことを特徴とする集積回路の開発方法を記録したプログラム記憶媒体。

５．　任意の規模、個数のブロックのポートとポートの接続情報からなるネットリストを生成する場合において、

トップ階層のポート名をネット名から接続先のブロックのポート名と一致するようにポート名を修正し、ブロックのポートとポートの接続情報からなるネットリストを生成することを特徴とする請求の範囲第２項に記載の集積回路の開発方法。

６．　ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、該グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成するネットリスト生成工程と、

前記ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成する論理合成工程と、

前記グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを前記ネットリスト生成工程により生成されたネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成するＲＯＭデータ生成工程と、

前記論理合成工程により作成されたＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を前記ＲＯＭデータ生成工程による実

機評価用ＲＯＭデータの生成とコンカレントに行なうレイアウト作成工程と、

前記ＲＯＭデータ生成工程により生成されたＲＯＭデータを用いた実機評価結果に基づく回路データの変更を前記ＡＳＩＣのレイアウト作成およびタイミング検証に反映する差分反映工程と、

を含んだことを特徴とするＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発方法。

７．　前記レイアウト作成工程は、前記ＲＯＭデータ生成工程より前か同時に始まり、前記差分反映工程は、前記レイアウト作成工程開始後におこなわれた前記回路データの変更をＡＳＩＣのレイアウト作成およびタイミング検証に反映することを特徴とする請求の範囲第６項に記載のＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発方法。

８．　前記ＲＯＭデータ生成工程は、前記グループ化した機能ブロックのうち前記論理合成工程によりプログラマブル・ロジックデバイス用論理合成データが作成されていない機能ブロックに対して、入力端子及び出力端子にダミー回路を挿入したプログラマブル・ロジックデバイス用論理合成データを前記ネットリスト生成工程により生成されたネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成する

ことを特徴とする請求の範囲第６項または第７項に記載のＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発方法。

９．　ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、該グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成するネットリスト生成手段と、

前記ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成デ

ータ及びプログラマブル・ロジックデバイス用論理合成データを作成する論理合成手段と、

前記グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを前記ネットリスト生成手段により生成されたネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成するＲＯＭデータ生成手段と、

前記論理合成手段により作成されたＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を前記ＲＯＭデータ生成手段による実機評価用ＲＯＭデータの生成とコンカレントに行なうレイアウト作成手段と、

を備えたことを特徴とするＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム。

１０．　ＡＳＩＣを構成する機能ブロックをポート間接続情報に基づいてグループ化し、該グループ化した機能ブロックのポートとポート間接続情報からなるネットリストをプログラマブル・ロジックデバイスのコア（論理コア）として生成するネットリスト生成手順と、

前記ＡＳＩＣを構成する機能ブロックの回路データからＡＳＩＣ用論理合成データ及びプログラマブル・ロジックデバイス用論理合成データを作成する論理合成手順と、

前記グループ化した機能ブロックについてのプログラマブル・ロジックデバイス用論理合成データを前記ネットリスト生成手順により生成されたネットリストに填め込んでプログラマブル・ロジックデバイスの回路を記録した実機評価用ＲＯＭデータを生成するＲＯＭデータ生成手順と、

前記論理合成手順により作成されたＡＳＩＣ用論理合成データを用いてＡＳＩＣのレイアウト作成およびタイミング検証を前記ＲＯＭデータ生成手順による実機評価用ＲＯＭデータの生成とコンカレントに行なうレイアウト作成手順と、

前記ＲＯＭデータ生成手順により生成されたＲＯＭデータを用いた実機評価結

果に基づく回路データの変更を前記ＡＳＩＣのレイアウト作成およびタイミング検証に反映する差分反映手順と、

をコンピュータに実行させることを特徴とするＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発プログラム。

１１．　ネットワークに接続されたコンピュータからユーザが利用するＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システムにおいて、

前記ユーザの要求に応じてＡＳＩＣの論理合成を実行するＡＳＩＣ論理合成手段と、

前記ＡＳＩＣ論理合成手段により作成されたＡＳＩＣの論理合成結果が前記ユーザの要求する速度性能を満足したか否かを判断するＡＳＩＣ論理合成結果判断手段と、

前記ＡＳＩＣ論理合成結果判断手段による判断結果に基づいてプログラマブル・ロジックデバイスの論理合成を実行するプログラマブル・ロジックデバイス論理合成手段と、

前記ＡＳＩＣ論理合成手段によるＡＳＩＣ論理合成の実行結果及び前記プログラマブル・ロジックデバイス論理合成手段によるプログラマブル・ロジックデバイス論理合成の実行結果を前記コンピュータに表示する論理合成結果表示手段と、

前記ＡＳＩＣ論理合成手段によるＡＳＩＣ論理合成の実行開始と実行結果及び前記プログラマブル・ロジックデバイス論理合成手段によるプログラマブル・ロジックデバイス論理合成の実行開始と実行結果を前記ユーザに電子メールで通知する論理合成通知手段と、

を備えたことを特徴とするＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム。

１２．　前記ユーザの要求に応じて前記ＡＳＩＣを構成する機能ブロックから前

記ユーザが指定する複数の機能ブロックのポート間接続情報から成るネットリストを生成するネットリスト生成手段と、前記ネットリスト生成手段により生成されたネットリストに論理合成済み対象機能ブロックのデータを填め込んでプログラマブル・ロジックデバイスの回路を記録したＲＯＭデータを生成するＲＯＭデータ生成手段と、前記ＲＯＭデータ生成手段により生成されたＲＯＭデータの生成結果を前記コンピュータに表示するＲＯＭデータ生成結果表示手段と、前記ＲＯＭデータ生成手段により生成されたＲＯＭデータの生成結果を前記ユーザに電子メールで通知するＲＯＭデータ生成結果通知手段とをさらに備えたことを特徴とする請求の範囲第１１項に記載のＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム。

１３．　前記ユーザが指定した前記ＡＳＩＣを構成する機能ブロックの設計が未完了で回路データが存在しない場合に、当該機能ブロックの入力端子及び出力端子にダミー回路を挿入したネットリストを生成する仮ネットリスト生成手段をさらに備えたことを特徴とする請求の範囲第１２項に記載のＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム。

１４．　前記ユーザが持つ最新の回路データとインプリメント設計者がインプリメント設計に取り込んでいる回路データ間の変更の規模を監視する監視手段と、前記監視手段による監視結果及びレイアウト設計にかかる時間に基づいて計画された日時に到達すると前記変更を前記ＡＳＩＣのインプリメント設計への反映のタイミングにあることを前記ユーザ及び前記ＡＳＩＣのインプリメント設計者に電子メールで通知する変更タイミング通知手段と、前記変更タイミング通知手段に応答して前記ユーザが前記反映の日付を変更することで停止を要求する反映停止要求手段とをさらに備えたことを特徴とする請求の範囲第１１項、第１２項または第１３項に記載のＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発システム。

１５．　ネットワークに接続されたコンピュータからユーザが利用するＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発方法において、

前記ユーザの要求に応じてＡＳＩＣの論理合成を実行するＡＳＩＣ論理合成工程と、

前記ＡＳＩＣ論理合成工程により作成されたＡＳＩＣの論理合成結果が前記ユーザの要求する速度性能を満足したか否かを判断するＡＳＩＣ論理合成結果判断工程と、

前記ＡＳＩＣ論理合成結果判断工程による判断結果に基づいてプログラマブル・ロジックデバイスの論理合成を実行するプログラマブル・ロジックデバイス論理合成工程と、

前記ＡＳＩＣ論理合成工程によるＡＳＩＣ論理合成の実行結果及び前記プログラマブル・ロジックデバイス論理合成工程によるプログラマブル・ロジックデバイス論理合成の実行結果を前記コンピュータに表示する論理合成結果表示工程と、

前記ＡＳＩＣ論理合成工程によるＡＳＩＣ論理合成の実行開始と実行結果及び前記プログラマブル・ロジックデバイス論理合成工程によるプログラマブル・ロジックデバイス論理合成の実行開始と実行結果を前記ユーザに電子メールで通知する論理合成通知工程と、

を含んだことを特徴とするＡＳＩＣとプログラマブル・ロジックデバイスのコンカレント開発方法。

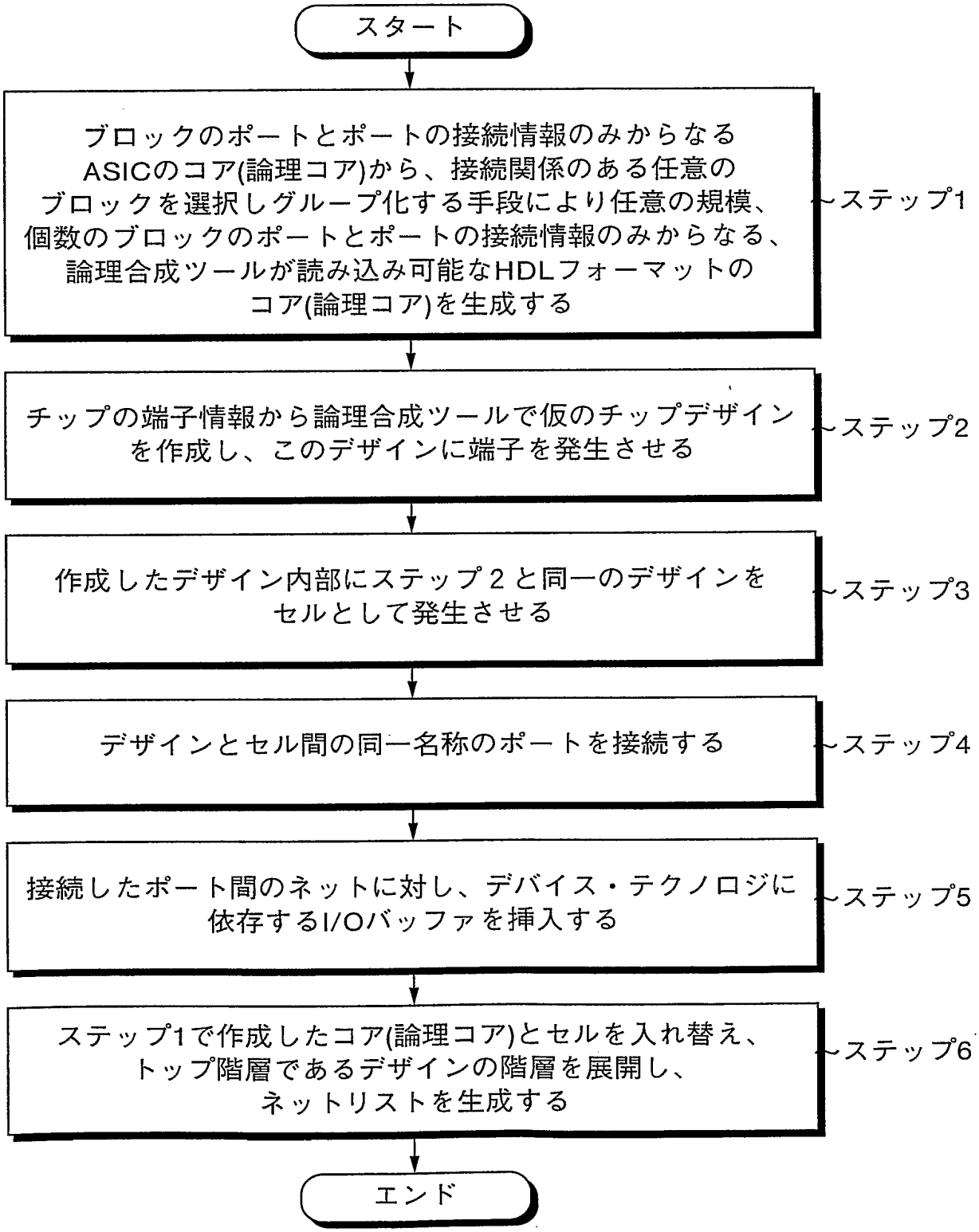
１６．　一方では、複数の機能ブロックの全部又は一部の機能ブロックを含むＦＰＧＡの端子情報が記述された第１デザインと、該第１デザインの下位層として記述された該ＦＰＧＡと同様の端子情報が記述された第２デザインとから、該第１デザイン、第２デザインにおける同一端子間が接続され、かつ、該端子間にＦＰＧＡ対応のバッファが挿入されたＦＰＧＡ用デザイン情報を生成し、

他方では、該複数の機能ブロックを含むＡＳＩＣの端子情報が記述された第３デザインと、該第３デザインの下位層として記述された該ＡＳＩＣと同様の端子情報が記述された第４デザインとから、該第３デザイン、第４デザイン間における同一端子間が接続され、かつ、該端子間にＡＳＩＣ対応のバッファが挿入されたＡＳＩＣ用デザイン情報を生成し、

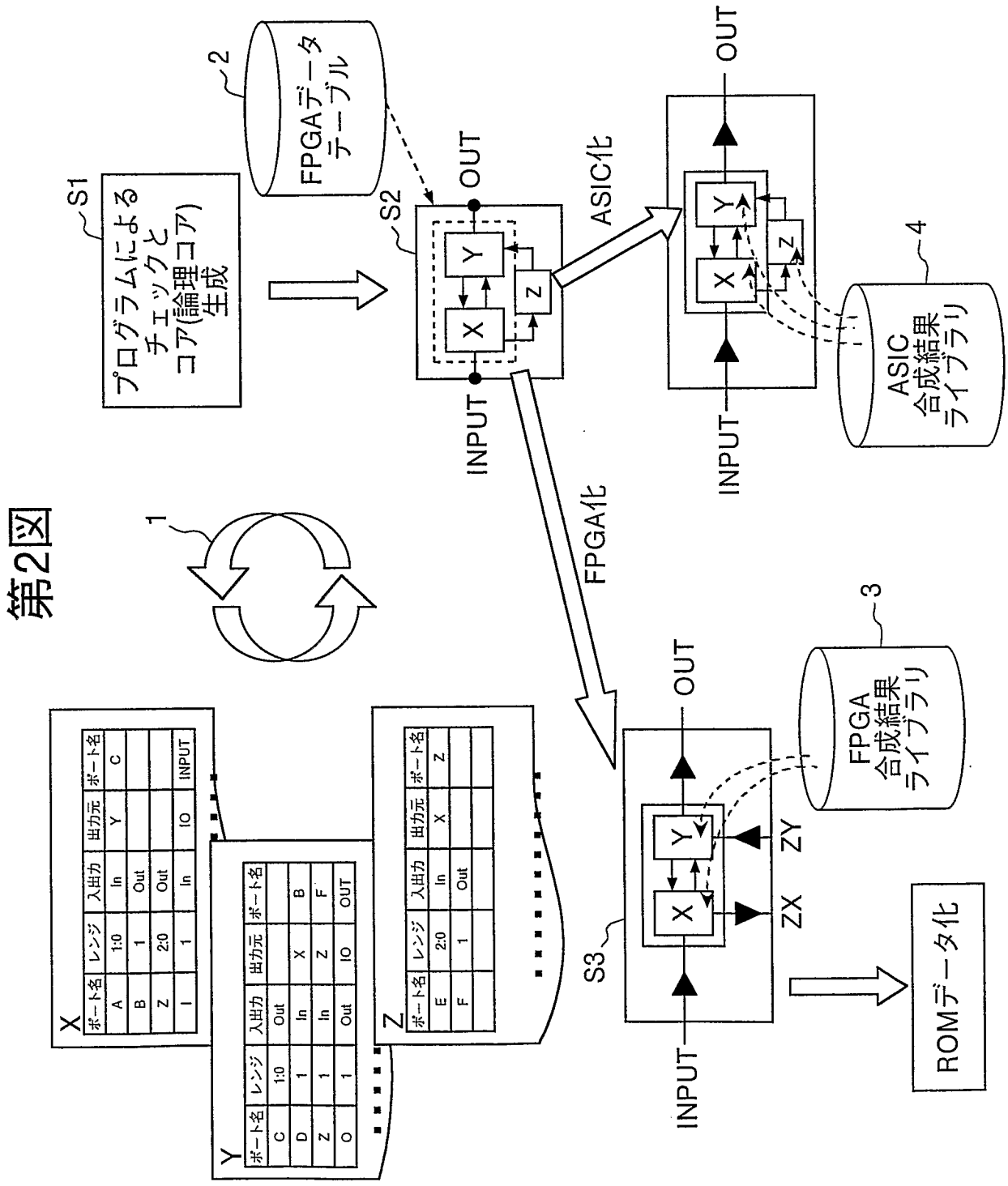
前記第２デザイン、第４デザインのそれぞれを、各デザインが含む機能ブロックの接続情報に基づいて生成された回路情報で置き換える、

ことでＦＰＧＡ及びＡＳＩＣのネットリストを生成する方法。

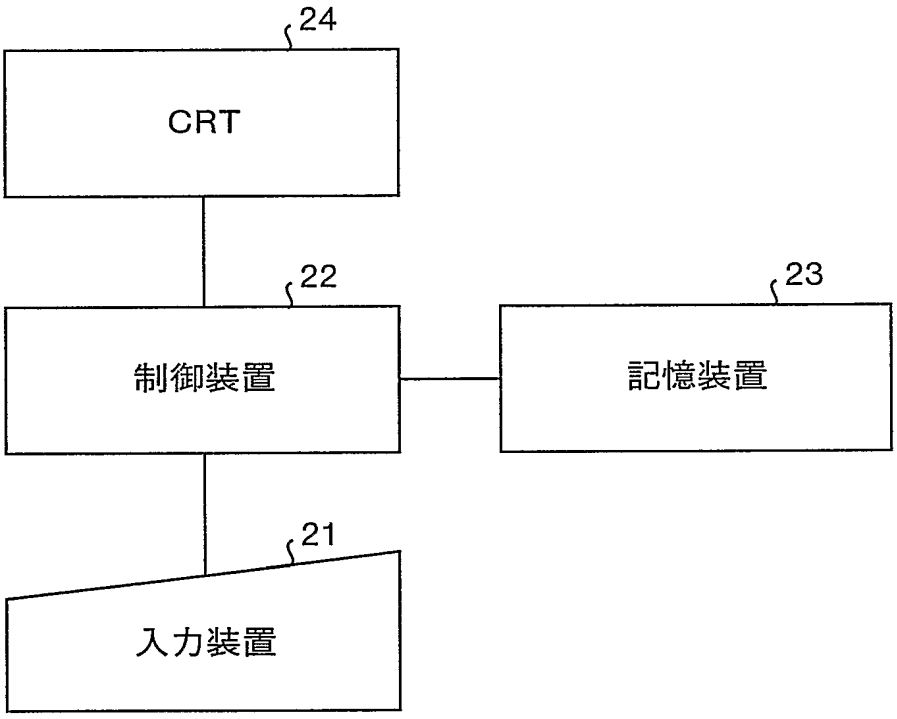
第1図


スタート
ブロックのポートとポートの接続情報のみからなるASICのコア(論理コア)から、接続関係のある任意のブロックを選択しグループ化する手段により任意の規模、個数のブロックのポートとポートの接続情報のみからなる、論理合成ツールが読み込み可能なHDLフォーマットのコア(論理コア)を生成する
ステップ1
チップの端子情報から論理合成ツールで仮のチップデザインを作成し、このデザインに端子を発生させる
ステップ2
作成したデザイン内部にステップ2と同一のデザインをセルとして発生させる
ステップ3
デザインとセル間の同一名称のポートを接続する
ステップ4
接続したポート間のネットに対し、デバイス・テクノロジに依存するI/Oバッファを挿入する
ステップ5
ステップ1で作成したコア(論理コア)とセルを入れ替え、トップ階層であるデザインの階層を展開し、ネットリストを生成する
ステップ6
エンド

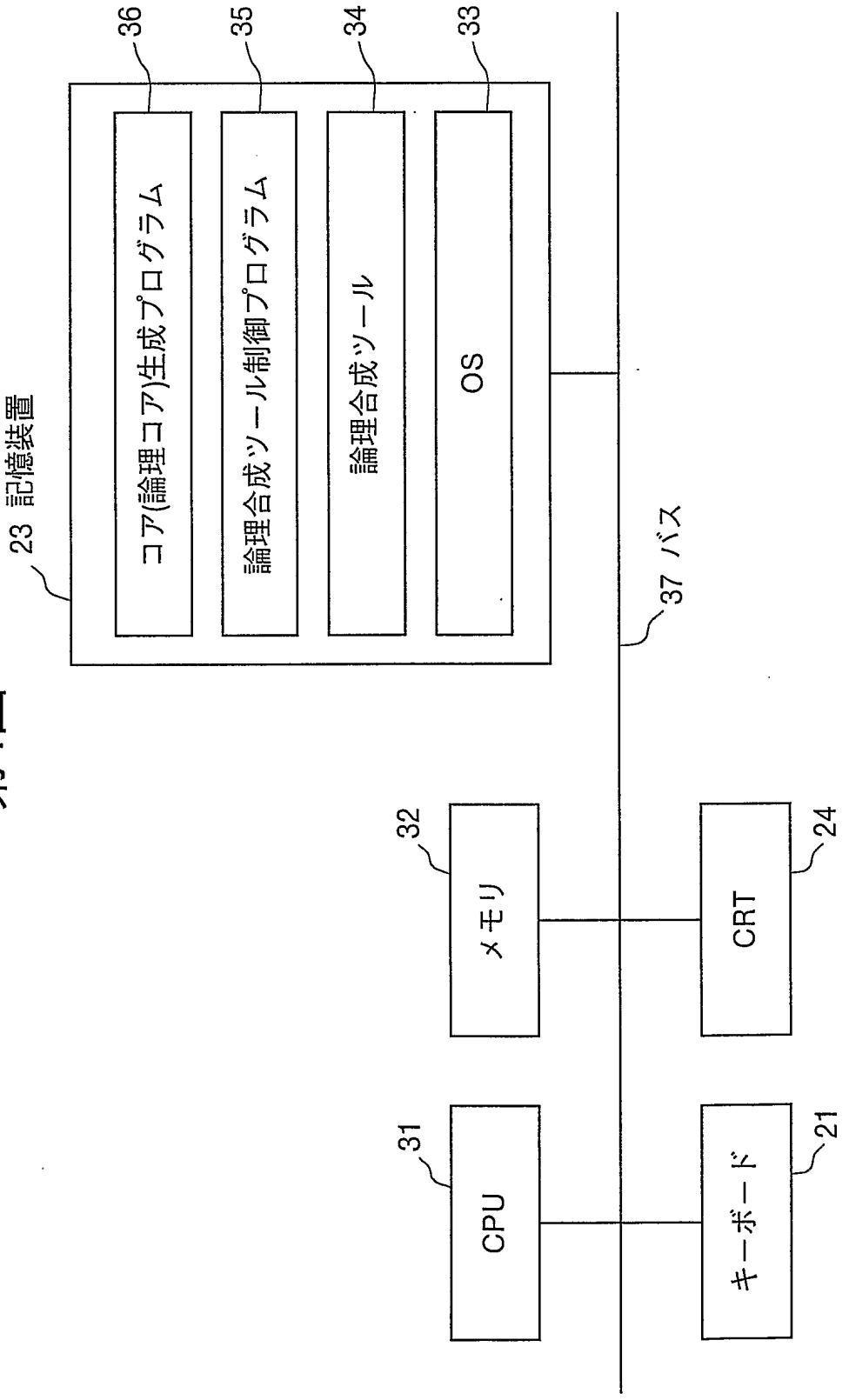
第2図


S1
プログラムによる
チェックと
コア(論理コア)
生成
2
FPGAデータ
テーブル
S2
OUT
INPUT
X
Y
Z
ASIC化
FPGA化
4
ASIC
合成結果
ライブラリ
1
3
FPGA
合成結果
ライブラリ
S3
ZX
ZY
ROMデータ化
X
ボート名 | レンジ | 入出力 | 出力元 | ボート名
A | 1:0 | In | Y | C
B | 1 | Out
Z | 2:0 | Out
I | 1 | In | IO | INPUT
Y
ボート名 | レンジ | 入出力 | 出力元 | ボート名
C | 1:0 | Out | |
D | 1 | In | X | B
Z | 1 | In | Z | F
O | 1 | Out | IO | OUT
Z
ボート名 | レンジ | 入出力 | 出力元 | ボート名
E | 2:0 | In | X | Z
F | 1 | Out

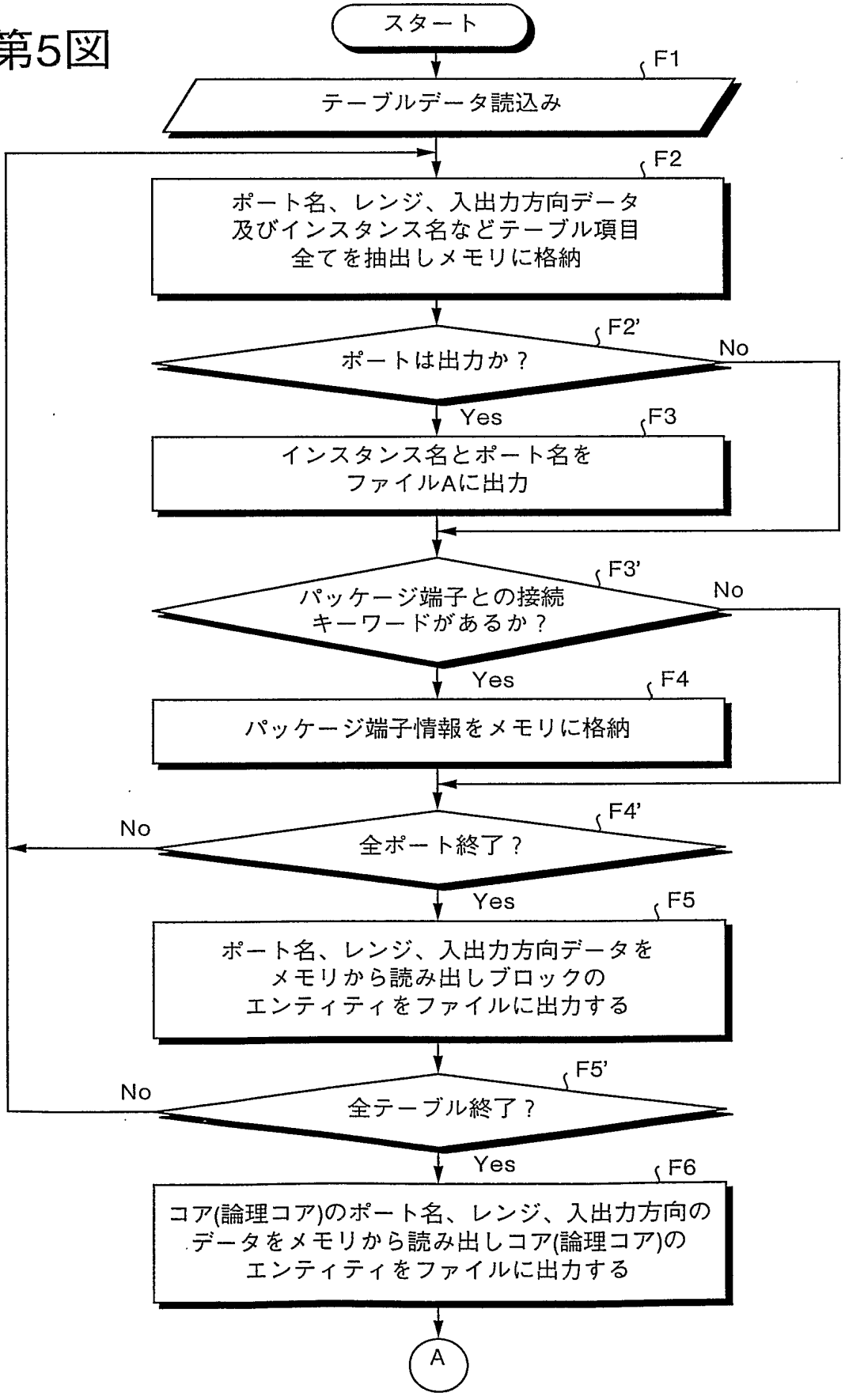
第3図



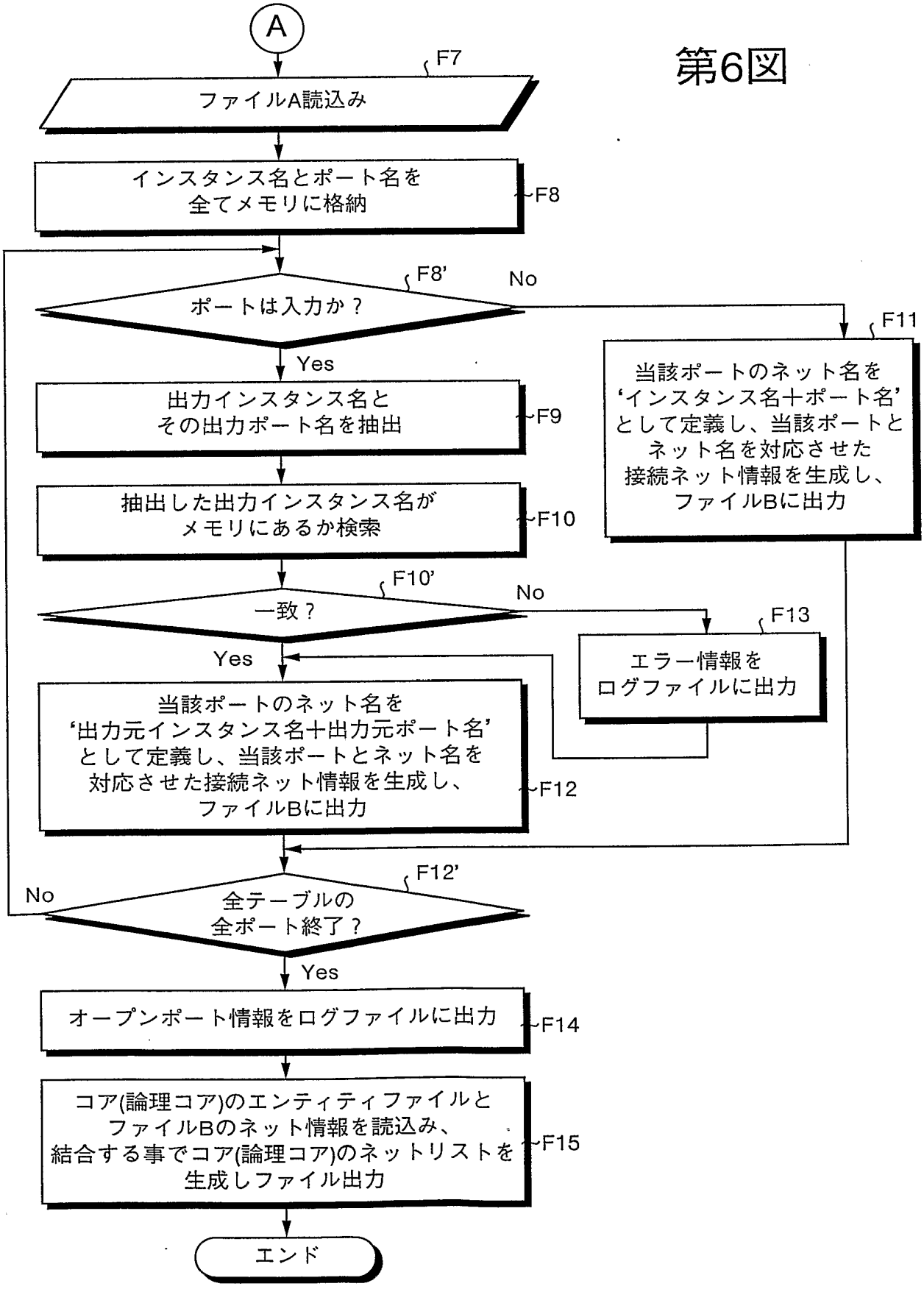
第4図


23 記憶装置
36
35
34
33
コア(論理コア)生成プログラム
論理合成ツール制御プログラム
論理合成ツール
OS
37 バス
32
メモリ
CRT
24
31
CPU
キーボード
21

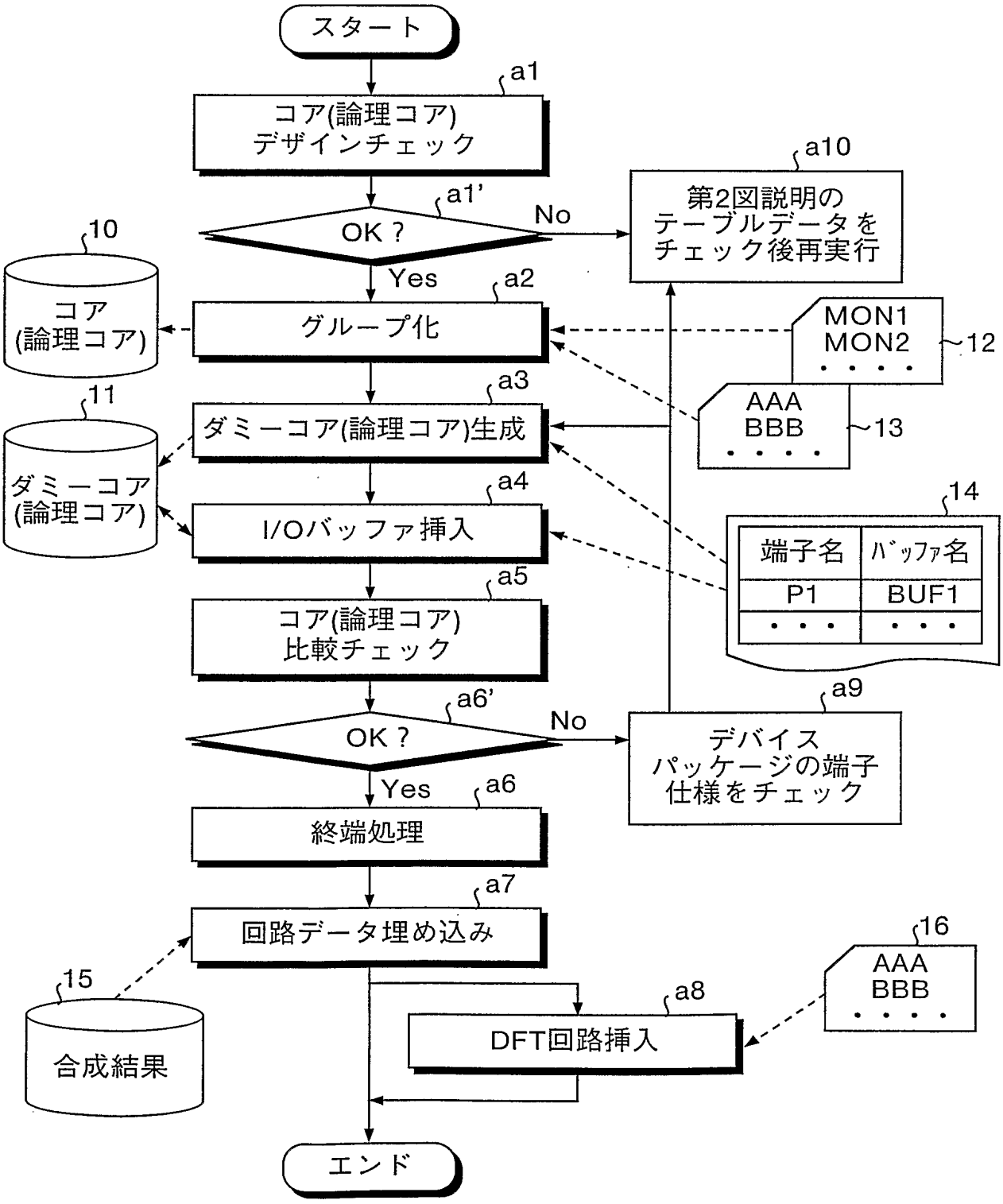
第5図


スタート
F1
テーブルデータ読込み
F2
ポート名、レンジ、入出力方向データ
及びインスタンス名などテーブル項目
全てを抽出しメモリに格納
F2'
ポートは出力か？
No
Yes
F3
インスタンス名とポート名を
ファイルAに出力
F3'
パッケージ端子との接続
キーワードがあるか？
No
Yes
F4
パッケージ端子情報をメモリに格納
F4'
No
全ポート終了？
Yes
F5
ポート名、レンジ、入出力方向データを
メモリから読み出しブロックの
エンティティをファイルに出力する
F5'
No
全テーブル終了？
Yes
F6
コア(論理コア)のポート名、レンジ、入出力方向の
データをメモリから読み出しコア(論理コア)の
エンティティをファイルに出力する
A

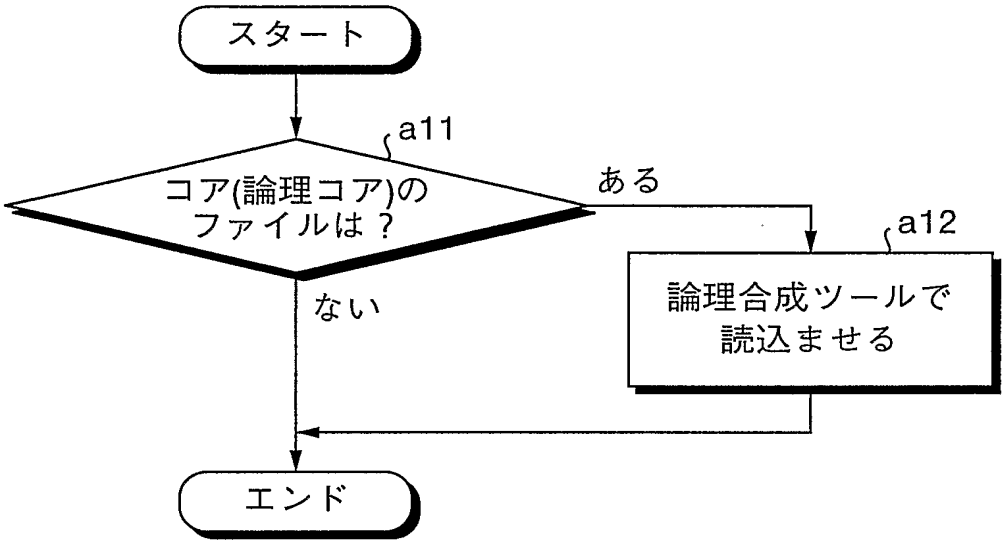
第6図


A
F7
ファイルA読込み
インスタンス名とポート名を
全てメモリに格納
F8
F8'
ポートは入力か？
No
Yes
F11
当該ポートのネット名を
'インスタンス名十ポート名'
として定義し、当該ポートと
ネット名を対応させた
接続ネット情報を生成し、
ファイルBに出力
出力インスタンス名と
その出力ポート名を抽出
F9
抽出した出力インスタンス名が
メモリにあるか検索
F10
F10'
一致？
No
F13
エラー情報を
ログファイルに出力
Yes
当該ポートのネット名を
'出力元インスタンス名十出力元ポート名'
として定義し、当該ポートとネット名を
対応させた接続ネット情報を生成し、
ファイルBに出力
F12
F12'
No
全テーブルの
全ポート終了？
Yes
オープンポート情報をログファイルに出力
F14
コア(論理コア)のエンティティファイルと
ファイルBのネット情報を読込み、
結合する事でコア(論理コア)のネットリストを
生成しファイル出力
F15
エンド

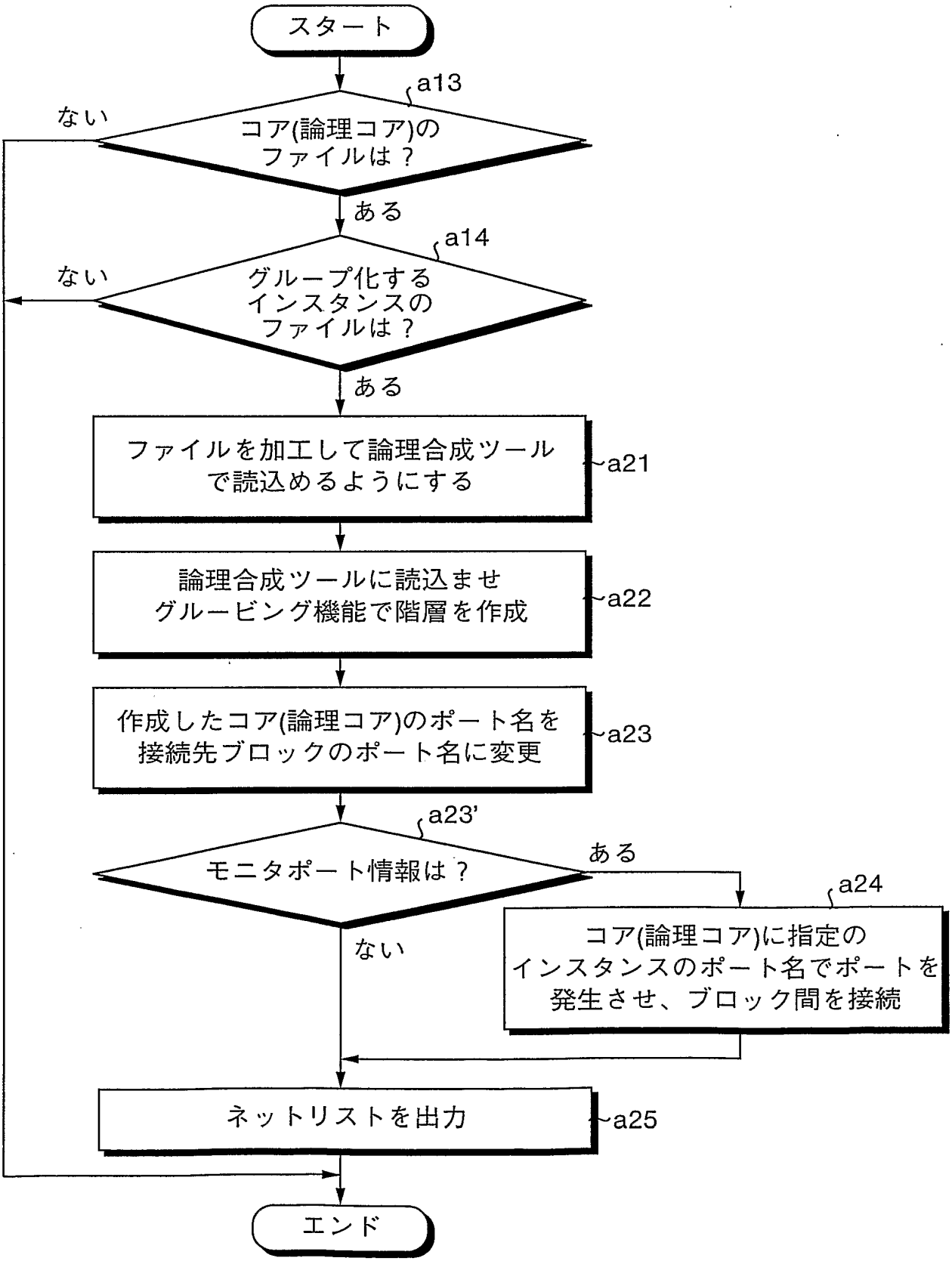
第7図


スタート
a1
コア(論理コア)
デザインチェック
a1'
OK ?
No
Yes
a10
第2図説明の
テーブルデータを
チェック後再実行
10
コア
(論理コア)
a2
グループ化
MON1
MON2
12
AAA
BBB
13
11
ダミーコア
(論理コア)
a3
ダミーコア(論理コア)生成
a4
I/Oバッファ挿入
14
端子名
ﾊﾞｯﾌｧ名
P1
BUF1
a5
コア(論理コア)
比較チェック
a6'
OK ?
No
Yes
a9
デバイス
パッケージの端子
仕様をチェック
a6
終端処理
a7
回路データ埋め込み
16
AAA
BBB
15
合成結果
a8
DFT回路挿入
エンド

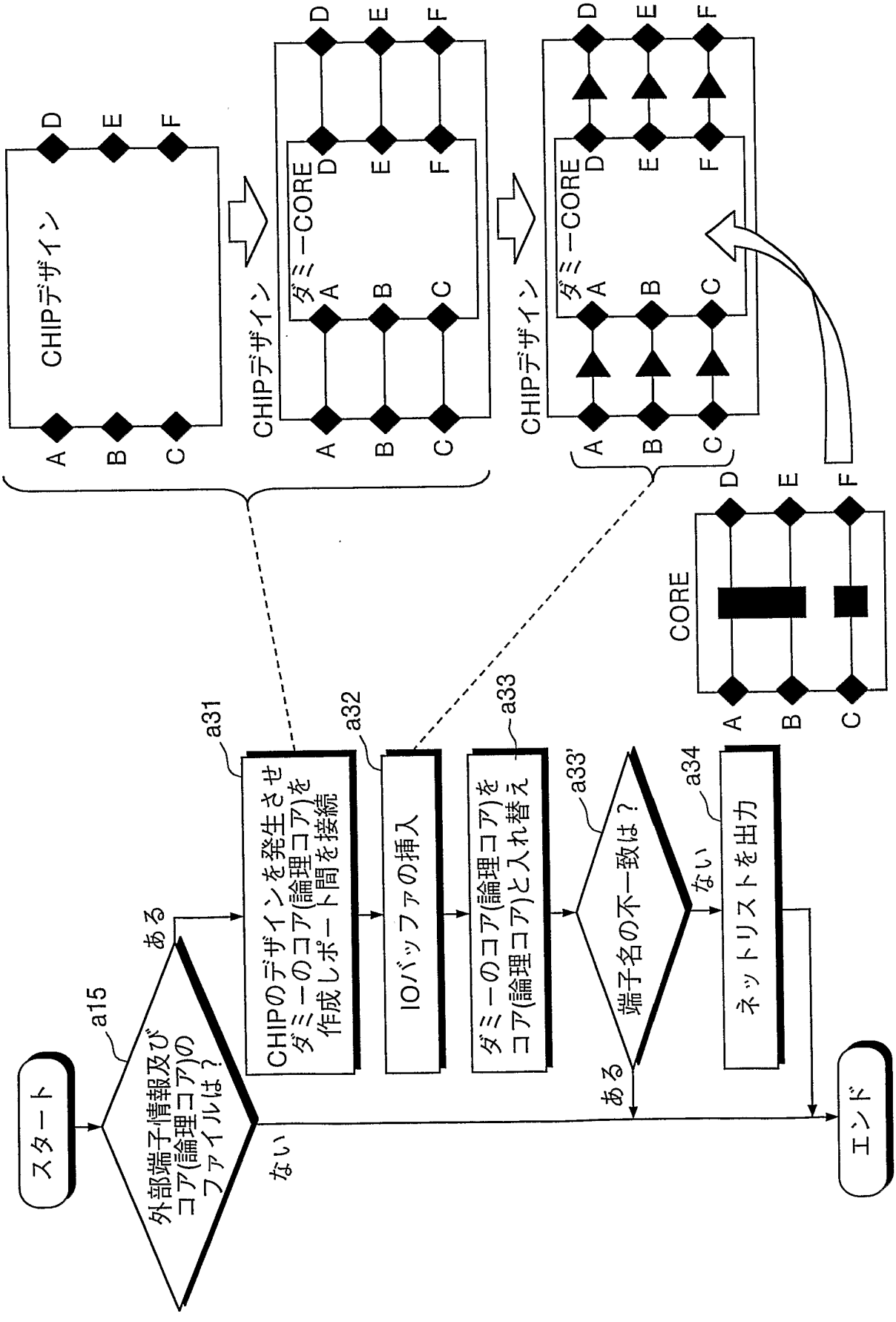
第8図



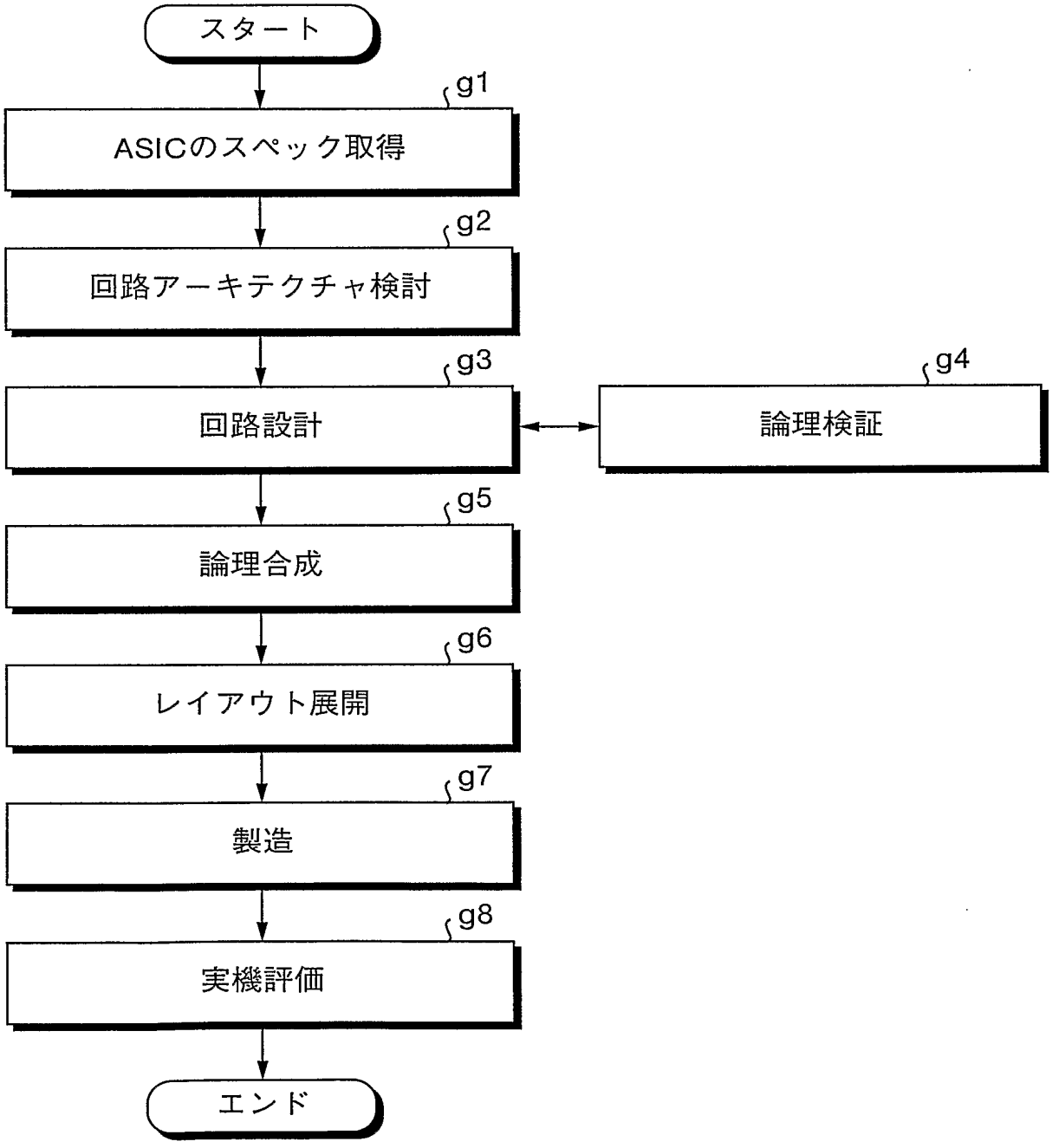
第9図


スタート
a13
コア(論理コア)の
ファイルは？
ない
ある
a14
グループ化する
インスタンスの
ファイルは？
ない
ある
ファイルを加工して論理合成ツール
で読込めるようにする
a21
論理合成ツールに読込ませ
グルービング機能で階層を作成
a22
作成したコア(論理コア)のポート名を
接続先ブロックのポート名に変更
a23
a23'
モニタポート情報は？
ある
ない
a24
コア(論理コア)に指定の
インスタンスのポート名でポートを
発生させ、ブロック間を接続
ネットリストを出力
a25
エンド

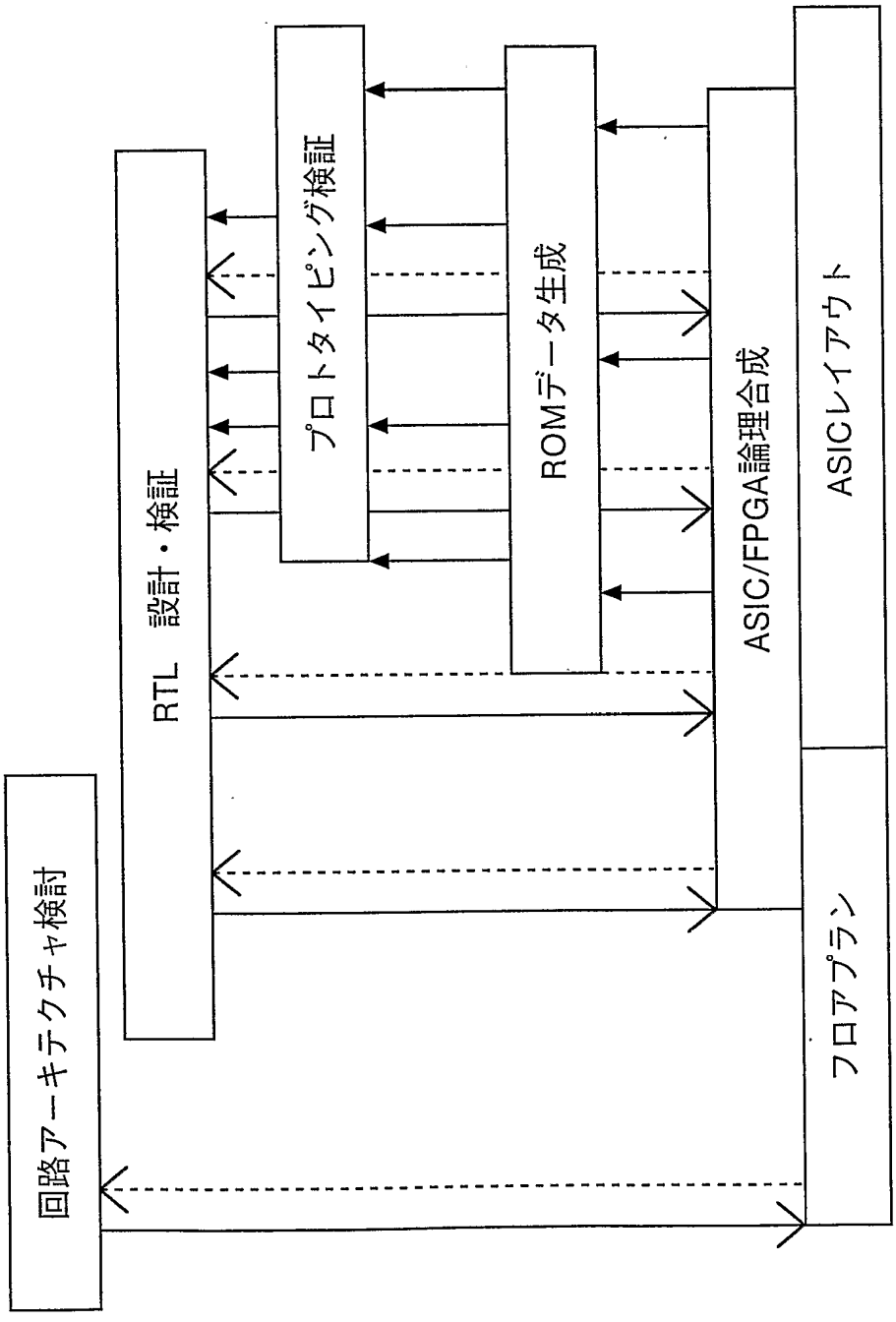
第10図


スタート
a15
外部端子情報及び
コア(論理コア)の
ファイルは？
ある
ない
a31
CHIPのデザインを発生させ
ダミーのコア(論理コア)を
作成しポート間を接続
a32
IOバッファの挿入
a33
ダミーのコア(論理コア)を
コア(論理コア)と入れ替え
a33'
端子名の不一致は？
ある
ない
a34
ネットリストを出力
エンド
CHIPデザイン
ダミーCORE
CORE
A
B
C
D
E
F

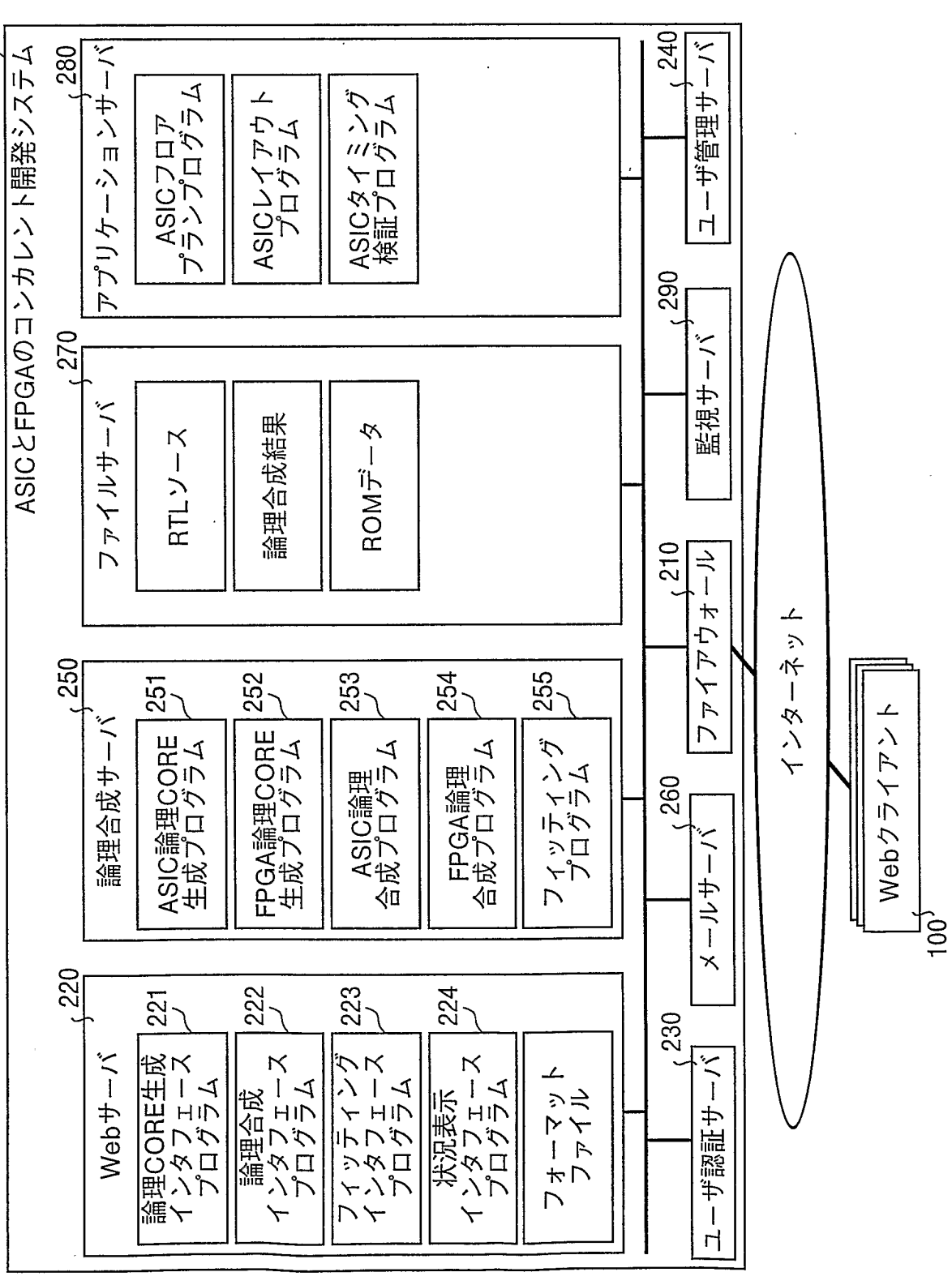
第11図


スタート
g1
ASICのスペック取得
g2
回路アーキテクチャ検討
g3
回路設計
g4
論理検証
g5
論理合成
g6
レイアウト展開
g7
製造
g8
実機評価
エンド

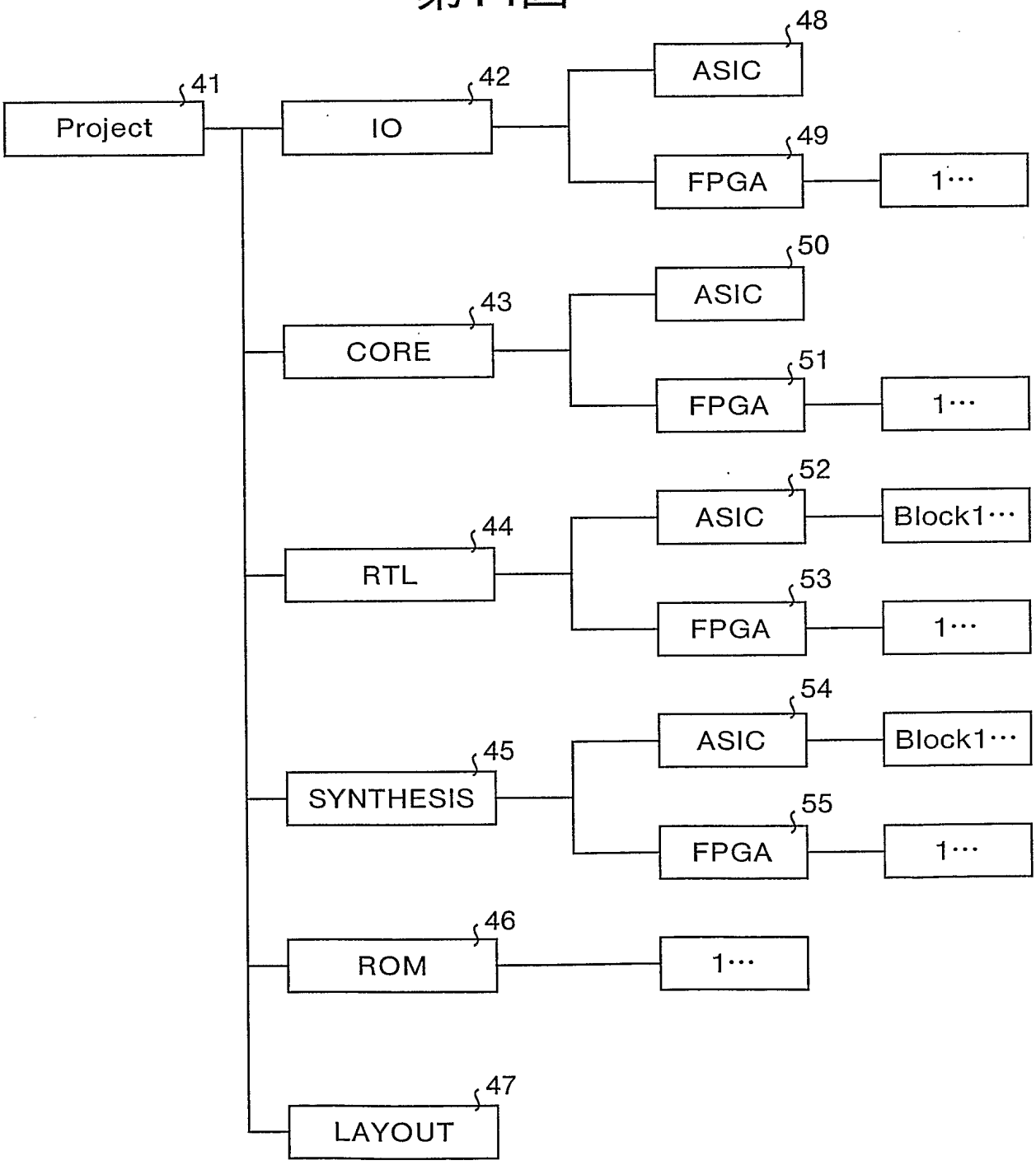
第12図


回路アーキテクチャ検討
RTL 設計・検証
プロトタイピング検証
ROMデータ生成
ASIC/FPGA論理合成
ASICレイアウト
フロアプラン

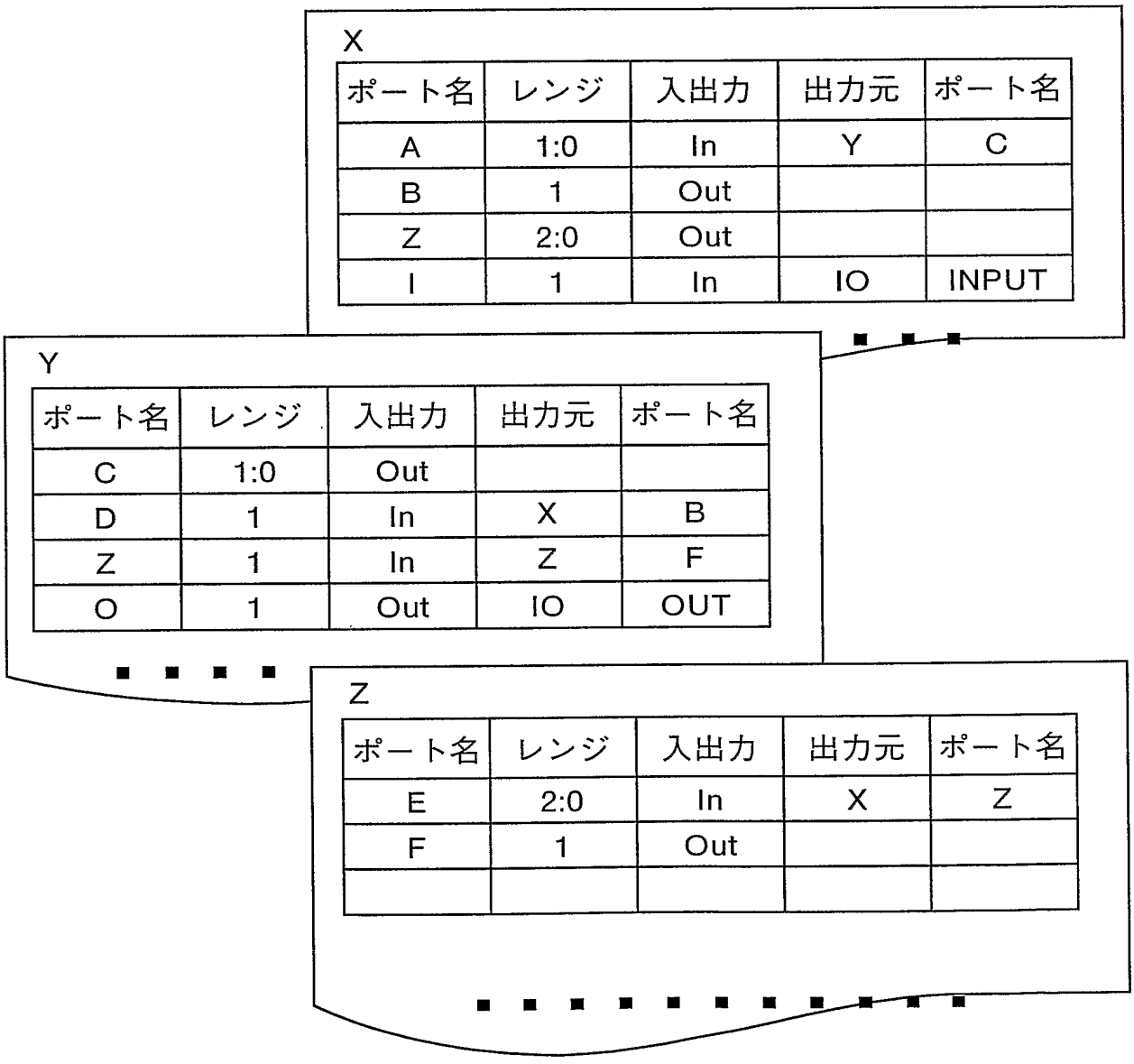
第13図


200
ASICとFPGAのコンカレント開発システム
220
Webサーバ
221
論理CORE生成インタフェースプログラム
222
論理合成インタフェースプログラム
223
フィッティングインタフェースプログラム
224
状況表示インタフェースプログラム
フォーマットファイル
250
論理合成サーバ
251
ASIC論理CORE生成プログラム
252
FPGA論理CORE生成プログラム
253
ASIC論理合成プログラム
254
FPGA論理合成プログラム
255
フィッティングプログラム
270
ファイルサーバ
RTLソース
論理合成結果
ROMデータ
280
アプリケーションサーバ
ASICフロアプランニングプログラム
ASICレイアウトプログラム
ASICタイミング検証プログラム
230
ユーザ認証サーバ
260
メールサーバ
210
ファイアウォール
290
監視サーバ
240
ユーザ管理サーバ
インターネット
Webクライアント
100

第14図


Project
41
IO
42
ASIC
48
FPGA
49
1…
CORE
43
ASIC
50
FPGA
51
1…
RTL
44
ASIC
52
Block1…
FPGA
53
1…
SYNTHESIS
45
ASIC
54
Block1…
FPGA
55
1…
ROM
46
1…
LAYOUT
47

第15図



X

ポート名	レンジ	入出力	出力元	ポート名
A	1:0	In	Y	C
B	1	Out		
Z	2:0	Out		
I	1	In	IO	INPUT

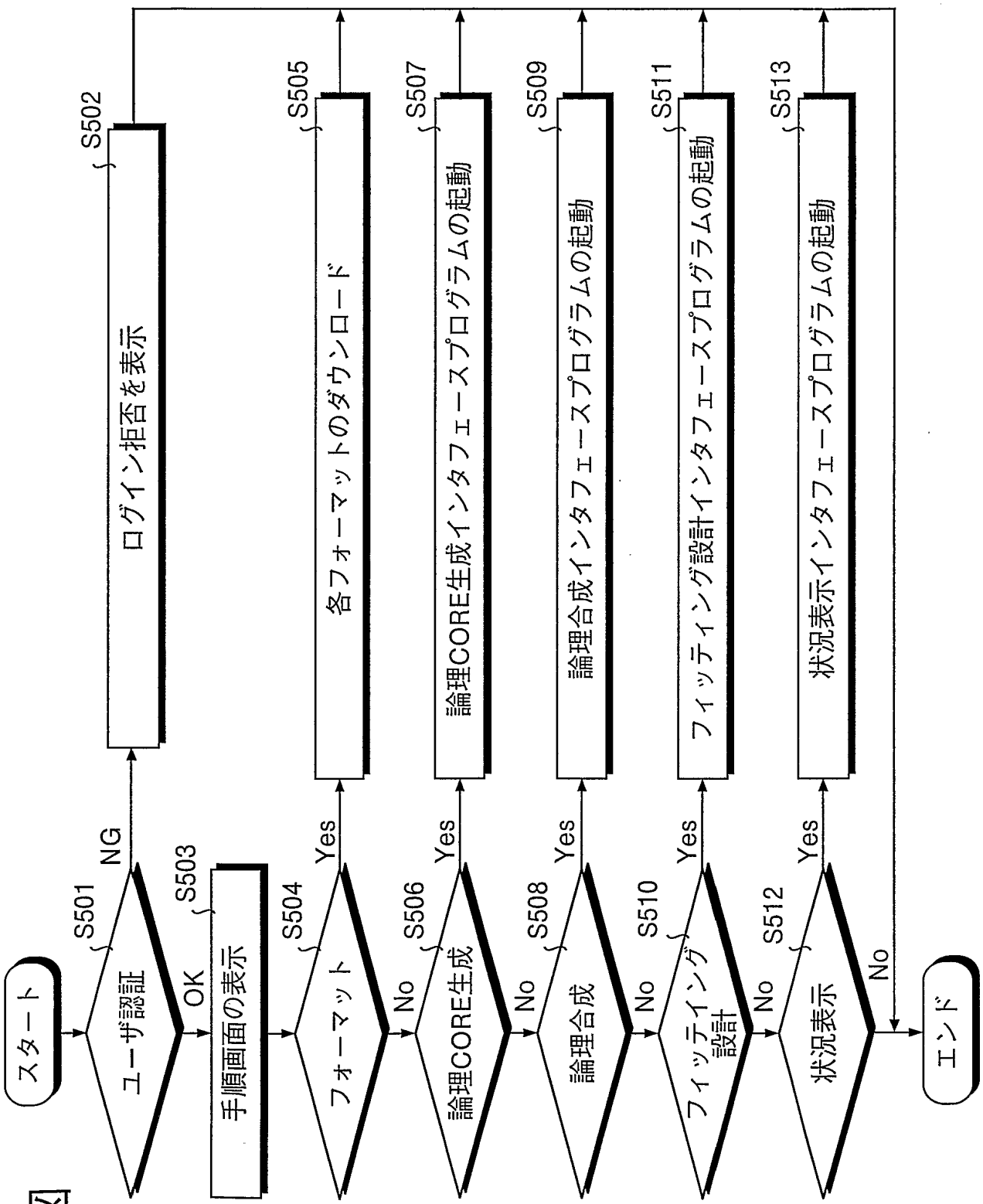
Y

ポート名	レンジ	入出力	出力元	ポート名
C	1:0	Out		
D	1	In	X	B
Z	1	In	Z	F
O	1	Out	IO	OUT

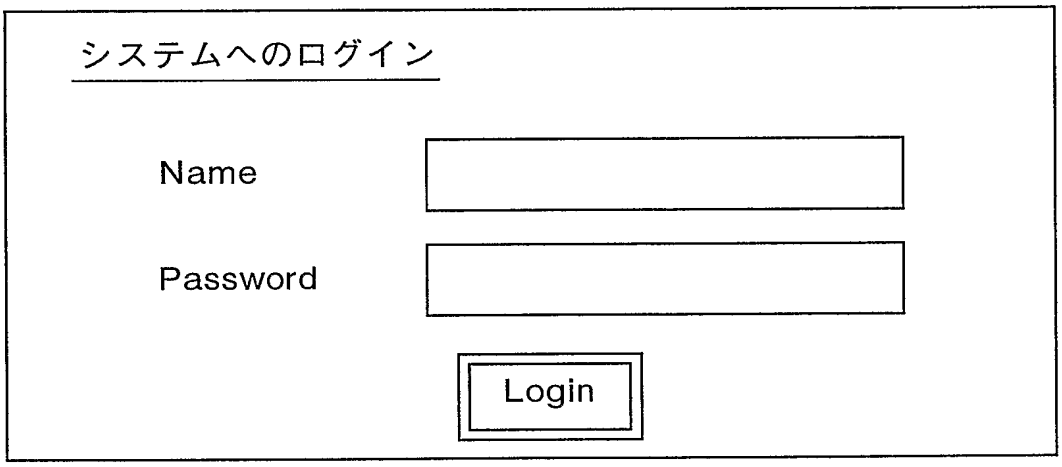
Z

ポート名	レンジ	入出力	出力元	ポート名
E	2:0	In	X	Z
F	1	Out		

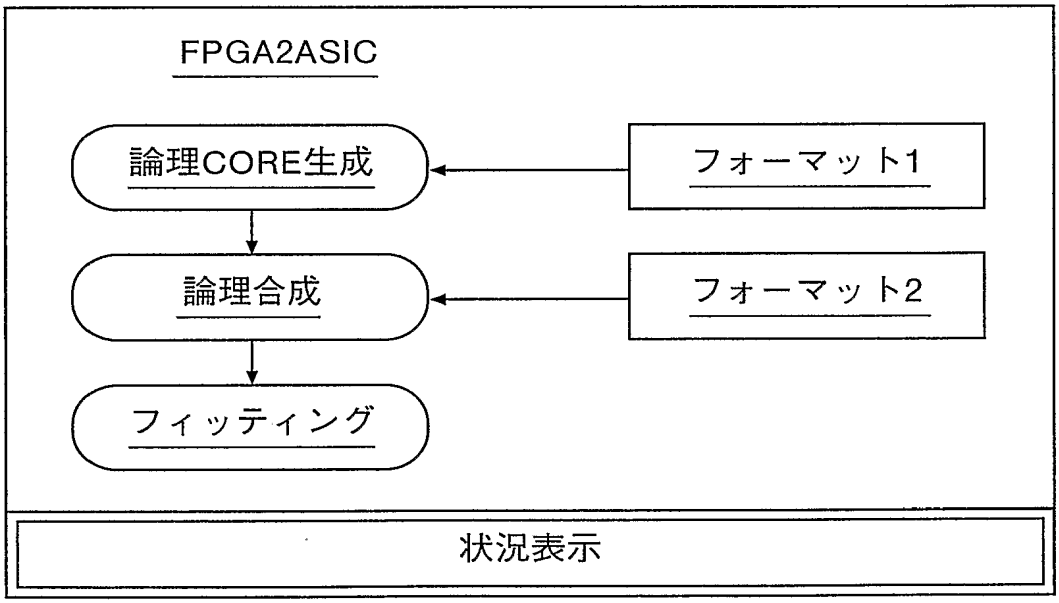
第16図


スタート
S501
ユーザ認証
NG
S502
ログイン拒否を表示
OK
S503
手順画面の表示
S504
フォーマット
Yes
S505
各フォーマットのダウンロード
No
S506
論理CORE生成
Yes
S507
論理CORE生成インタフェースプログラムの起動
No
S508
論理合成
Yes
S509
論理合成インタフェースプログラムの起動
No
S510
フィッティング設計
Yes
S511
フィッティング設計インタフェースプログラムの起動
No
S512
状況表示
Yes
S513
状況表示インタフェースプログラムの起動
No
エンド

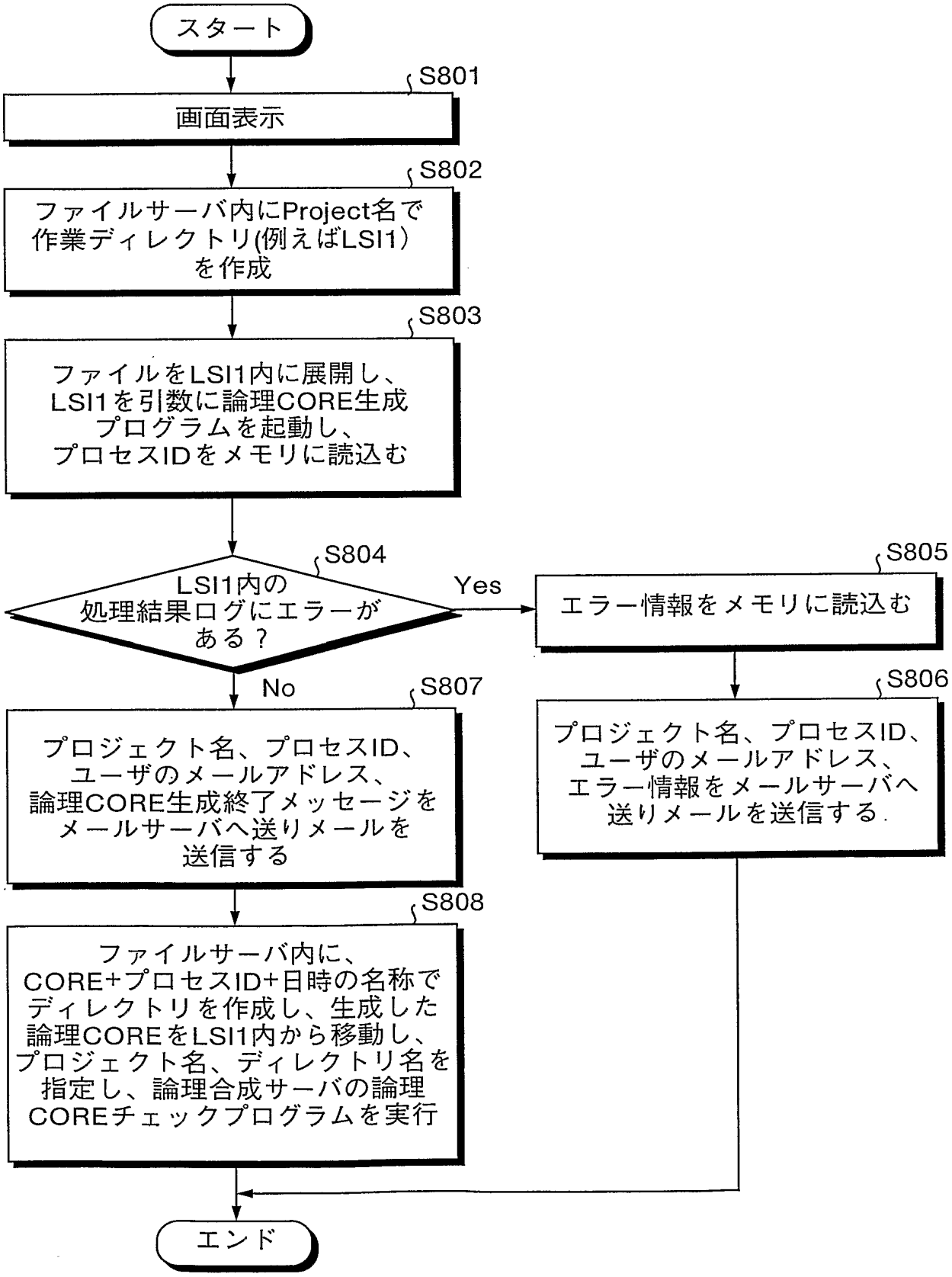
第17図



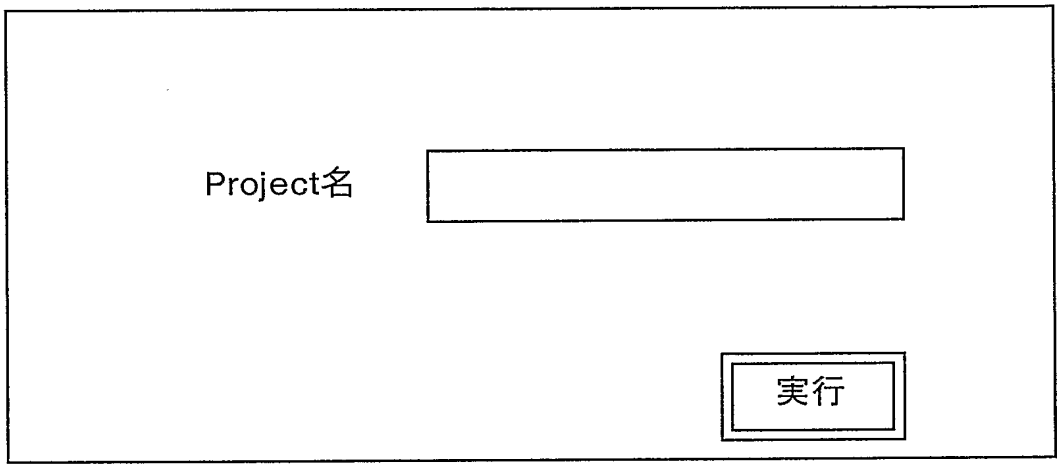
第18図



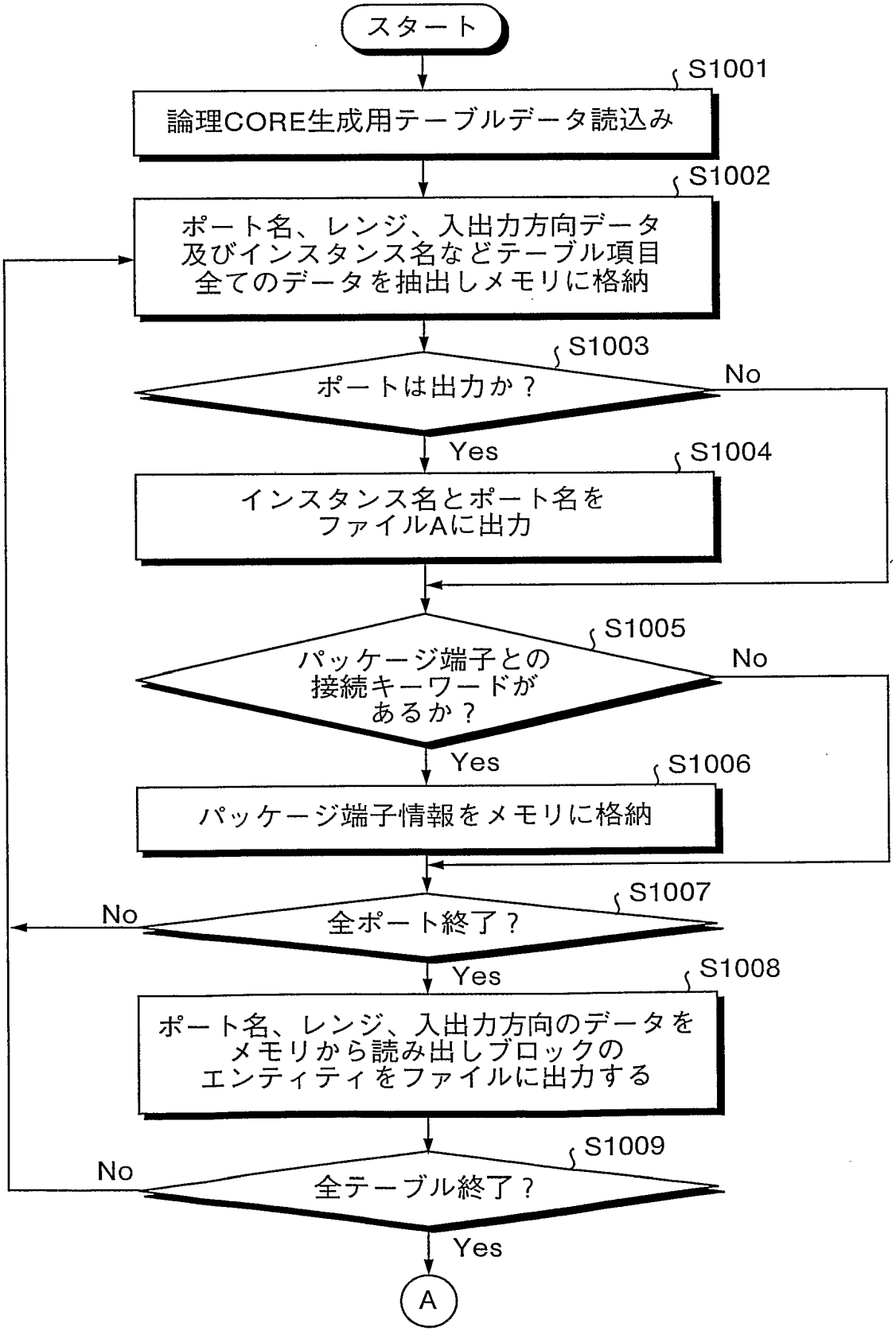
第19図


スタート
S801
画面表示
S802
ファイルサーバ内にProject名で作業ディレクトリ(例えばLSI1)を作成
S803
ファイルをLSI1内に展開し、LSI1を引数に論理CORE生成プログラムを起動し、プロセスIDをメモリに読込む
S804
LSI1内の処理結果ログにエラーがある？
Yes
S805
エラー情報をメモリに読込む
No
S807
プロジェクト名、プロセスID、ユーザのメールアドレス、論理CORE生成終了メッセージをメールサーバへ送りメールを送信する
S806
プロジェクト名、プロセスID、ユーザのメールアドレス、エラー情報をメールサーバへ送りメールを送信する
S808
ファイルサーバ内に、CORE+プロセスID+日時の名称でディレクトリを作成し、生成した論理COREをLSI1内から移動し、プロジェクト名、ディレクトリ名を指定し、論理合成サーバの論理COREチェックプログラムを実行
エンド

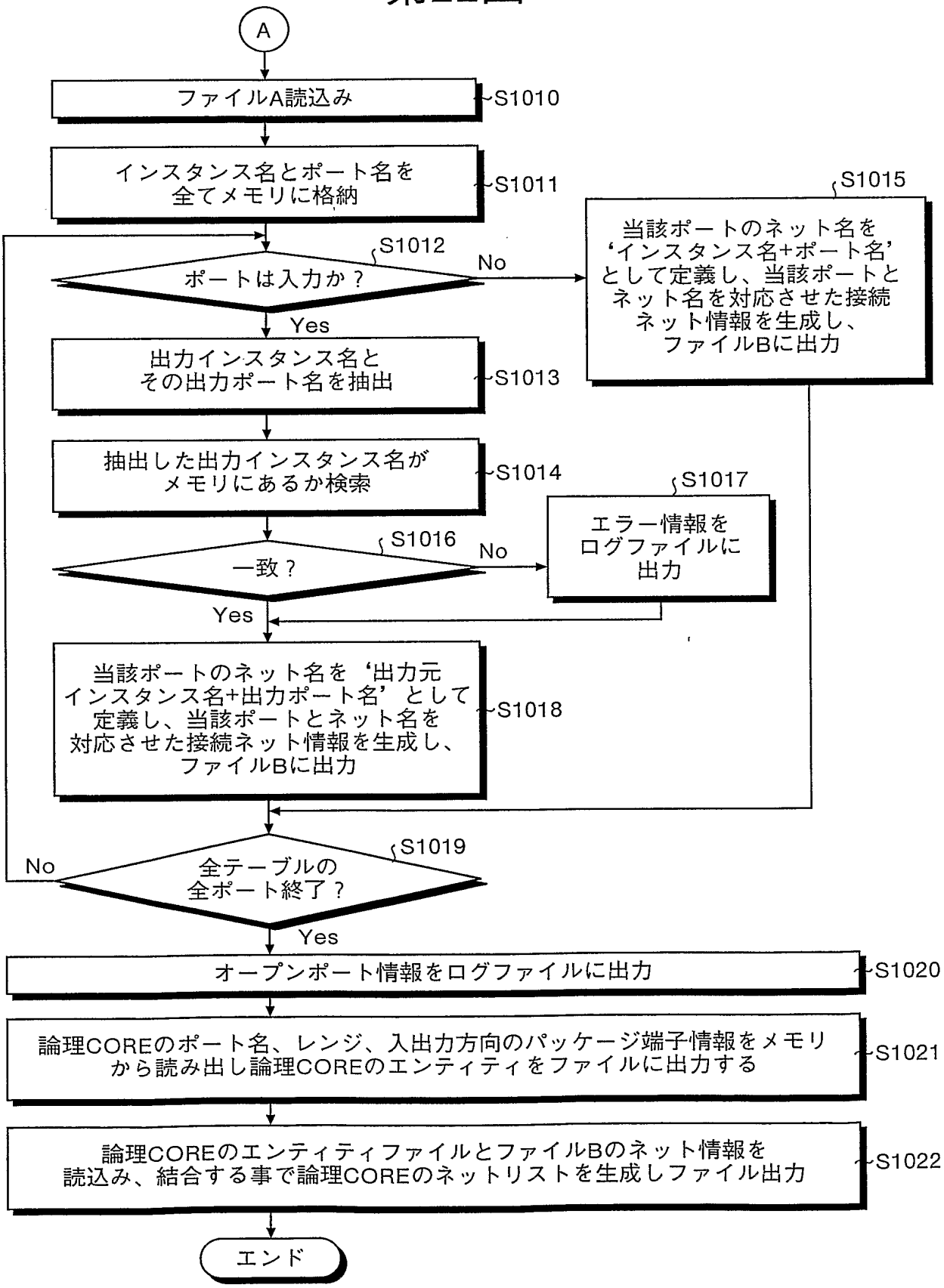
第20図



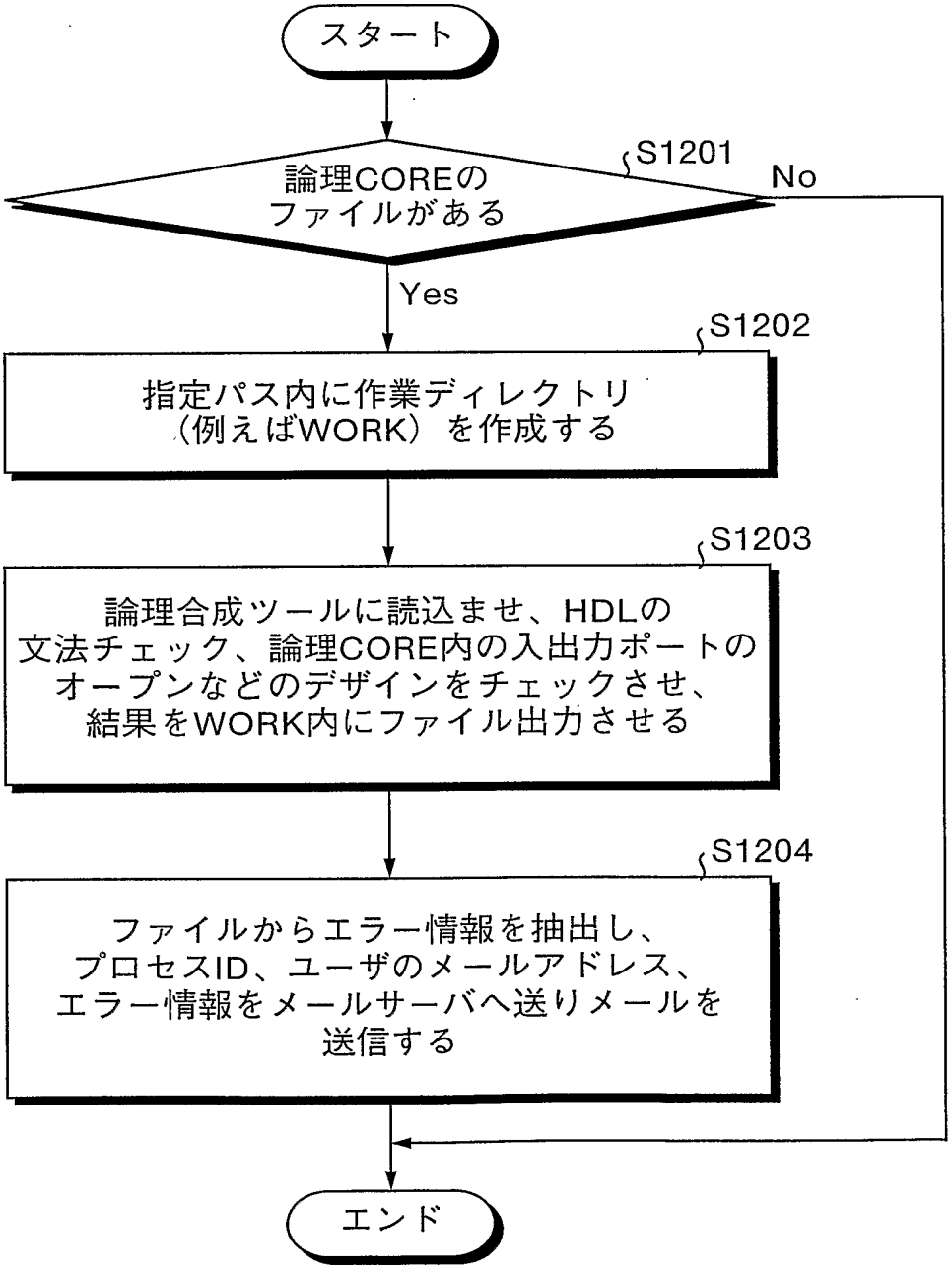
第21図


スタート
S1001
論理CORE生成用テーブルデータ読込み
S1002
ポート名、レンジ、入出力方向データ
及びインスタンス名などテーブル項目
全てのデータを抽出しメモリに格納
S1003
ポートは出力か？
No
Yes
S1004
インスタンス名とポート名を
ファイルAに出力
S1005
パッケージ端子との
接続キーワードが
あるか？
No
Yes
S1006
パッケージ端子情報をメモリに格納
S1007
No
全ポート終了？
Yes
S1008
ポート名、レンジ、入出力方向のデータを
メモリから読み出しブロックの
エンティティをファイルに出力する
S1009
No
全テーブル終了？
Yes
A

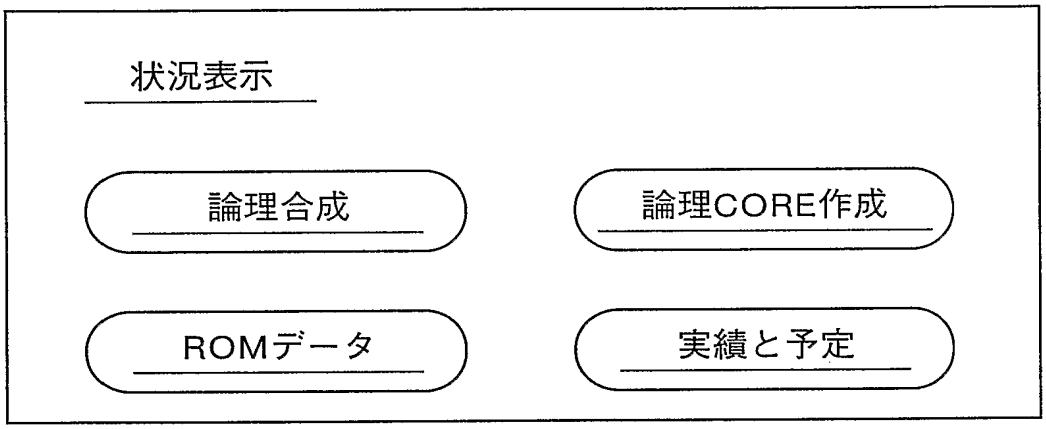
第22図


A
ファイルA読込み
S1010
インスタンス名とポート名を
全てメモリに格納
S1011
S1012
ポートは入力か？
No
Yes
S1015
当該ポートのネット名を
'インスタンス名+ポート名'
として定義し、当該ポートと
ネット名を対応させた接続
ネット情報を生成し、
ファイルBに出力
出力インスタンス名と
その出力ポート名を抽出
S1013
抽出した出力インスタンス名が
メモリにあるか検索
S1014
S1017
エラー情報を
ログファイルに
出力
S1016
一致？
No
Yes
当該ポートのネット名を'出力元
インスタンス名+出力ポート名'として
定義し、当該ポートとネット名を
対応させた接続ネット情報を生成し、
ファイルBに出力
S1018
S1019
全テーブルの
全ポート終了？
No
Yes
オープンポート情報をログファイルに出力
S1020
論理COREのポート名、レンジ、入出力方向のパッケージ端子情報をメモリ
から読み出し論理COREのエンティティをファイルに出力する
S1021
論理COREのエンティティファイルとファイルBのネット情報を
読込み、結合する事で論理COREのネットリストを生成しファイル出力
S1022
エンド

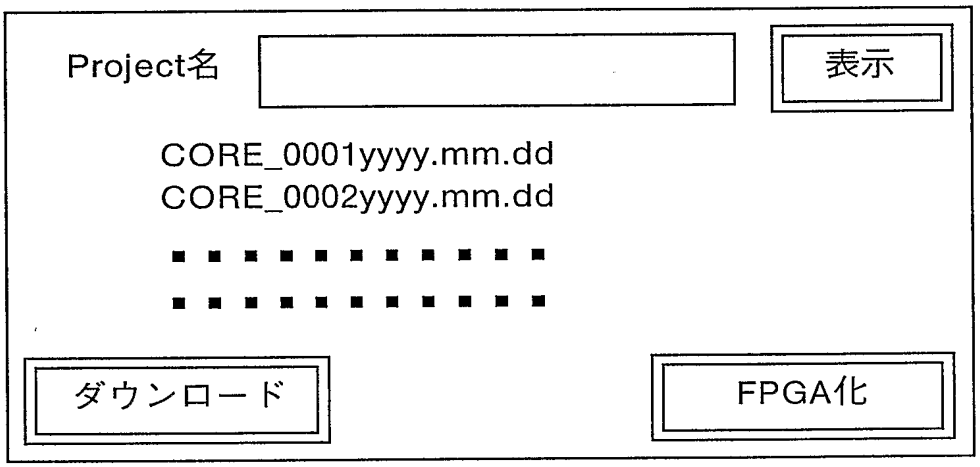
第23図


スタート
論理COREの
ファイルがある
S1201
No
Yes
S1202
指定パス内に作業ディレクトリ
（例えばWORK）を作成する
S1203
論理合成ツールに読込ませ、HDLの
文法チェック、論理CORE内の入出力ポートの
オープンなどのデザインをチェックさせ、
結果をWORK内にファイル出力させる
S1204
ファイルからエラー情報を抽出し、
プロセスID、ユーザのメールアドレス、
エラー情報をメールサーバへ送りメールを
送信する
エンド

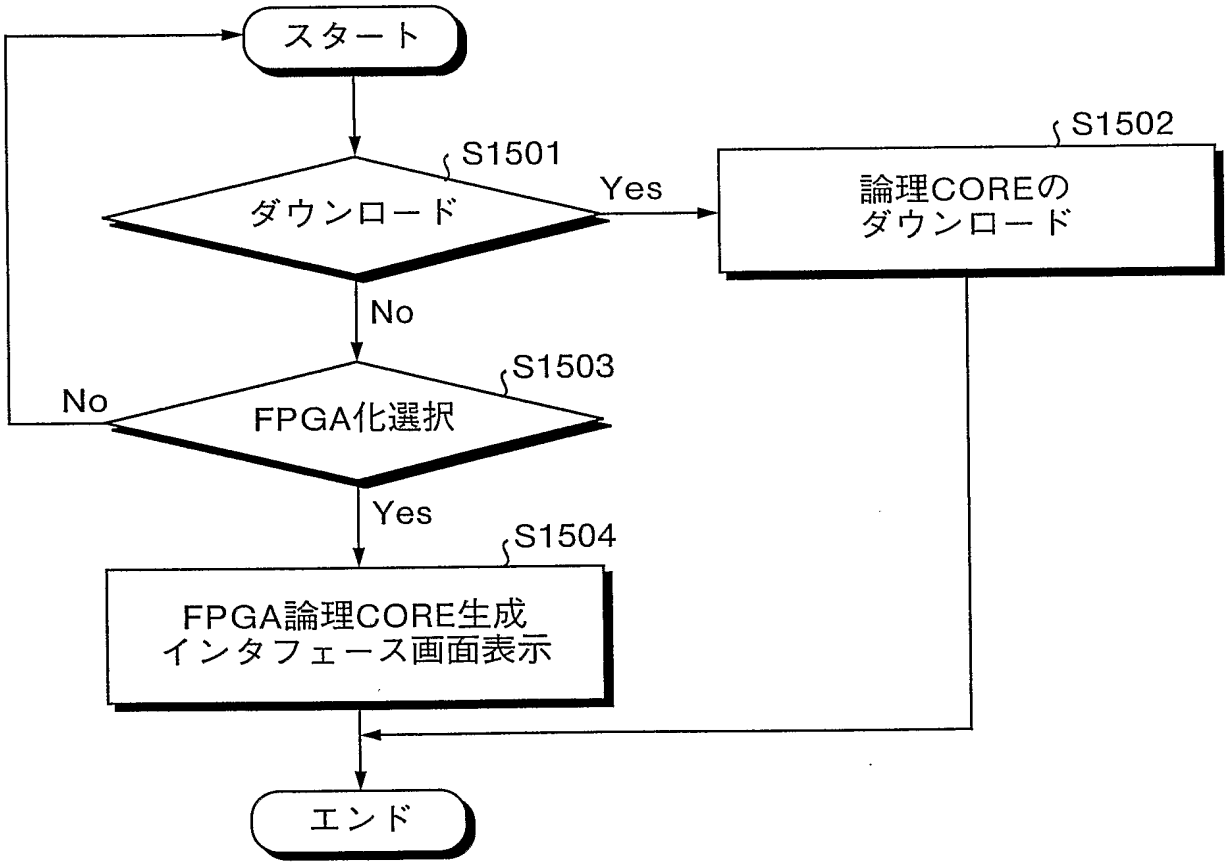
第24図



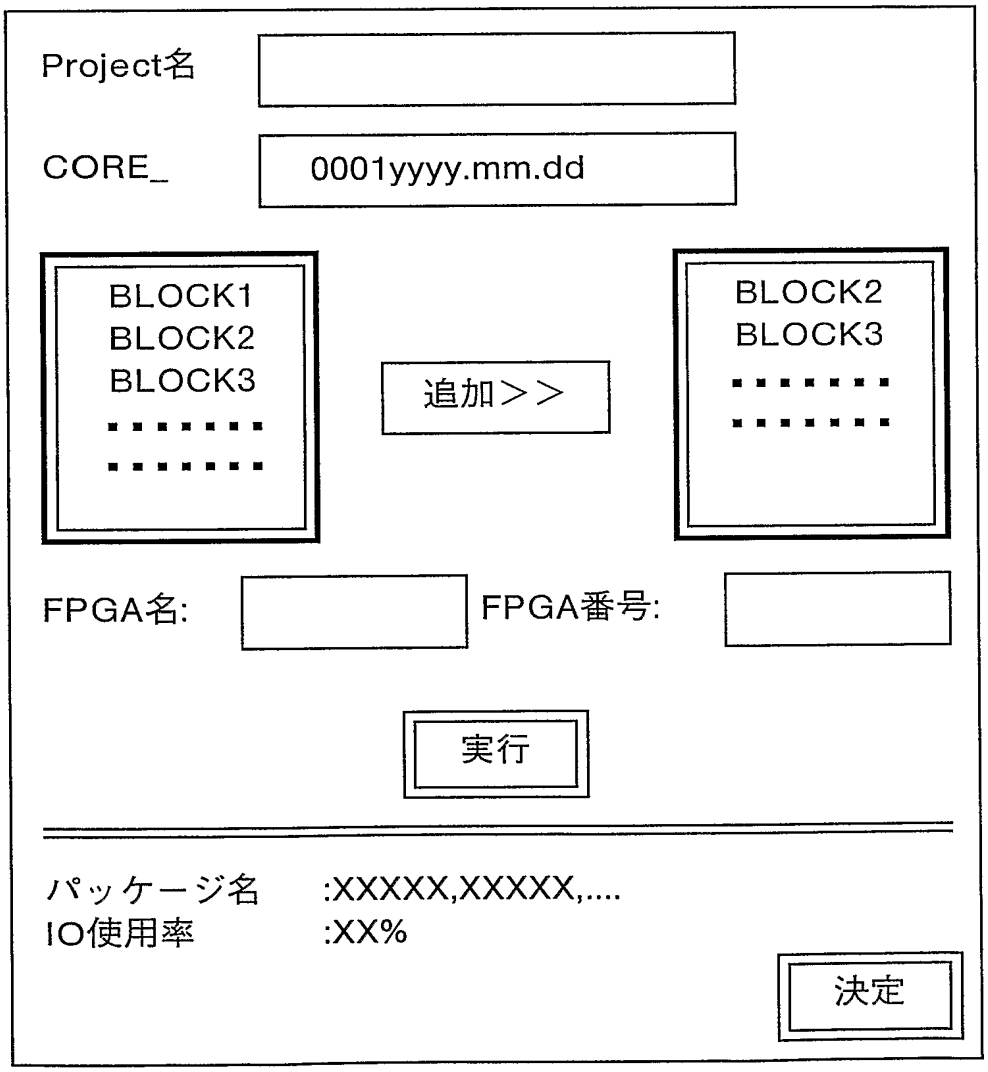
第25図


Project名
表示
CORE_0001yyyy.mm.dd
CORE_0002yyyy.mm.dd
ダウンロード
FPGA化

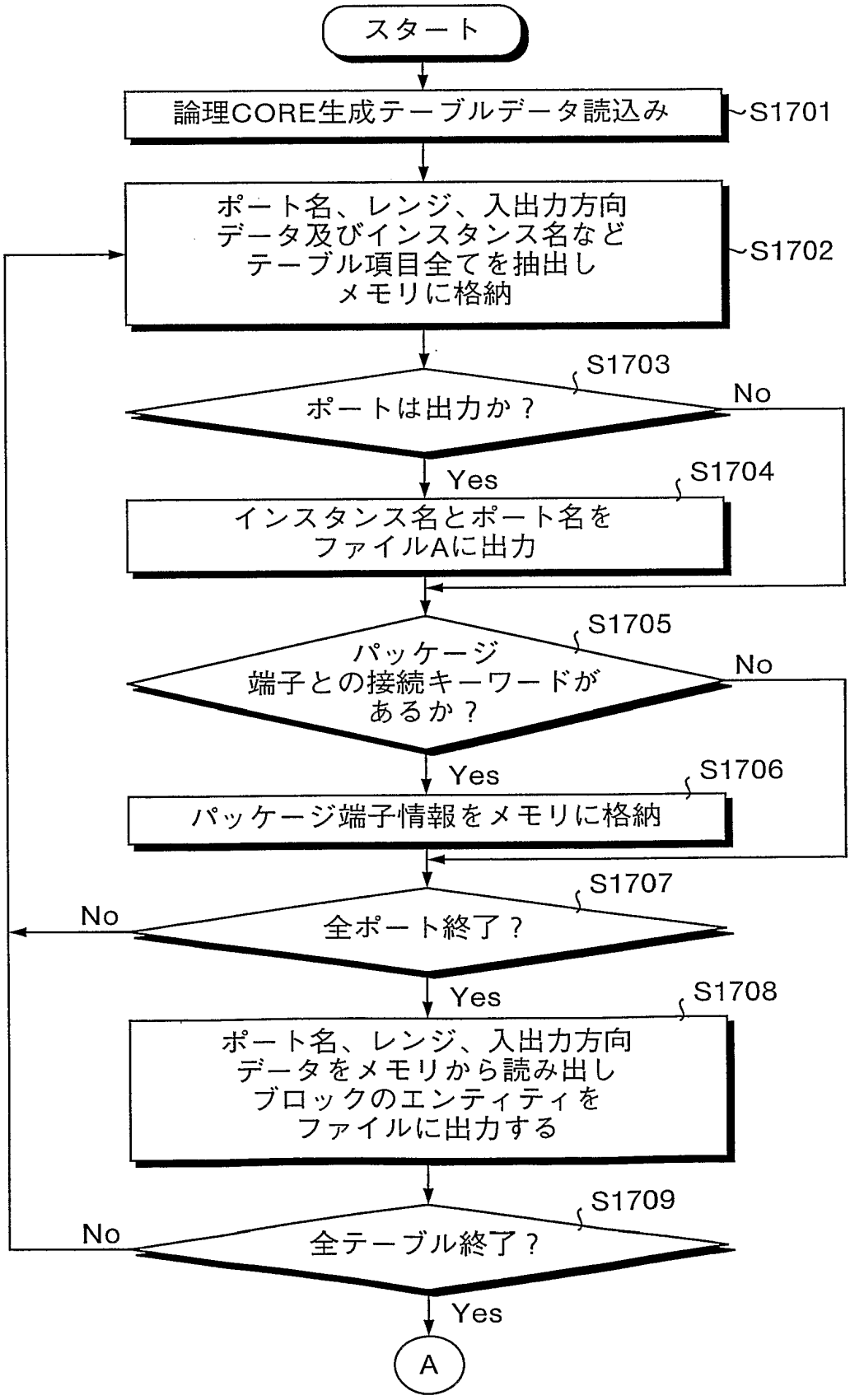
第26図


スタート
S1501
ダウンロード
Yes
S1502
論理COREの
ダウンロード
No
S1503
No
FPGA化選択
Yes
S1504
FPGA論理CORE生成
インタフェース画面表示
エンド

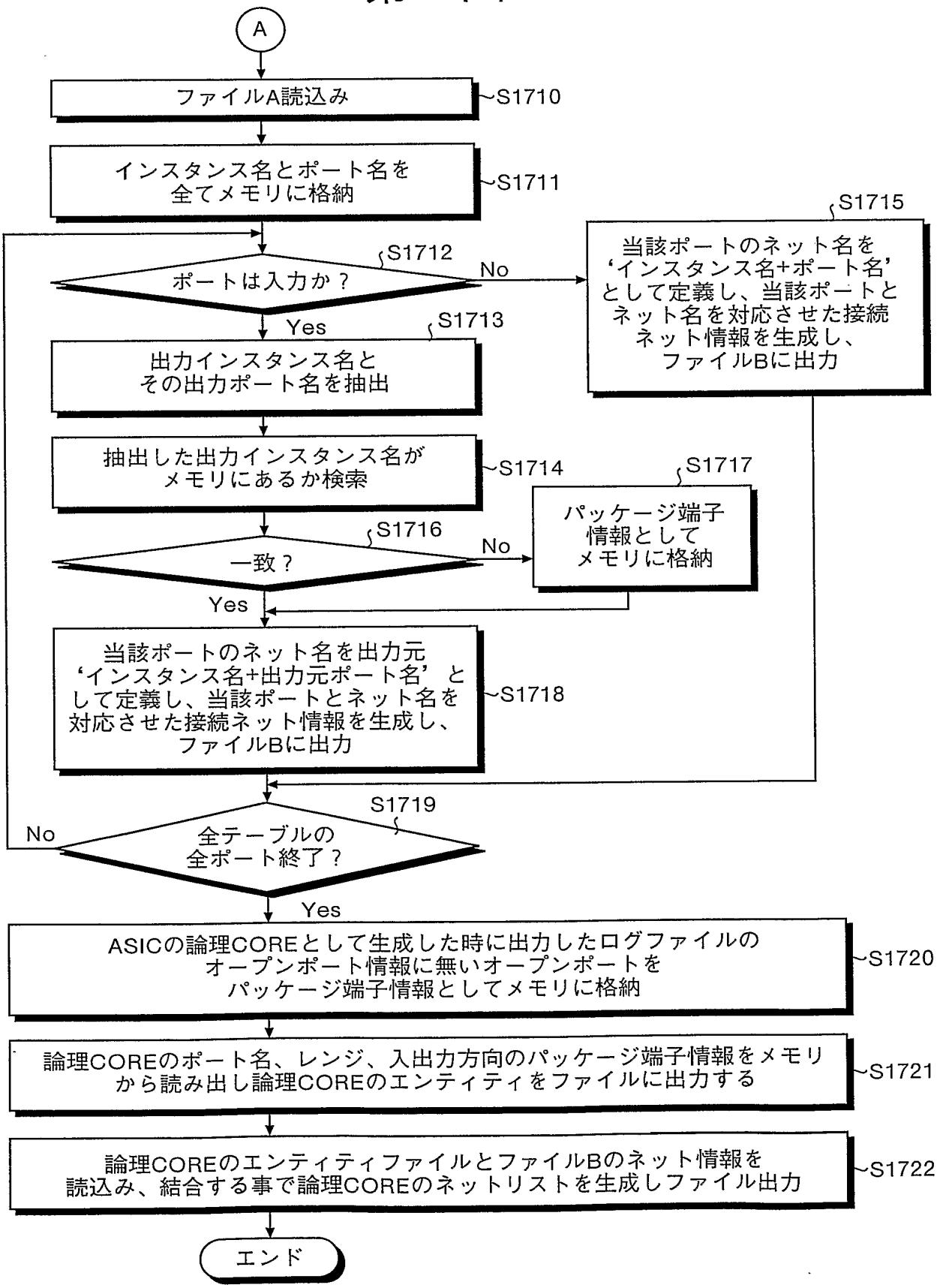
第27図


Project名
CORE_
0001yyyy.mm.dd
BLOCK1
BLOCK2
BLOCK3
追加＞＞
BLOCK2
BLOCK3
FPGA名:
FPGA番号:
実行
パッケージ名 :XXXXX,XXXXX,....
IO使用率 :XX%
決定

第28図


スタート
論理CORE生成テーブルデータ読込み
S1701
ポート名、レンジ、入出力方向
データ及びインスタンス名など
テーブル項目全てを抽出し
メモリに格納
S1702
S1703
ポートは出力か？
No
Yes
S1704
インスタンス名とポート名を
ファイルAに出力
S1705
パッケージ
端子との接続キーワードが
あるか？
No
Yes
S1706
パッケージ端子情報をメモリに格納
S1707
No
全ポート終了？
Yes
S1708
ポート名、レンジ、入出力方向
データをメモリから読み出し
ブロックのエンティティを
ファイルに出力する
S1709
No
全テーブル終了？
Yes
A

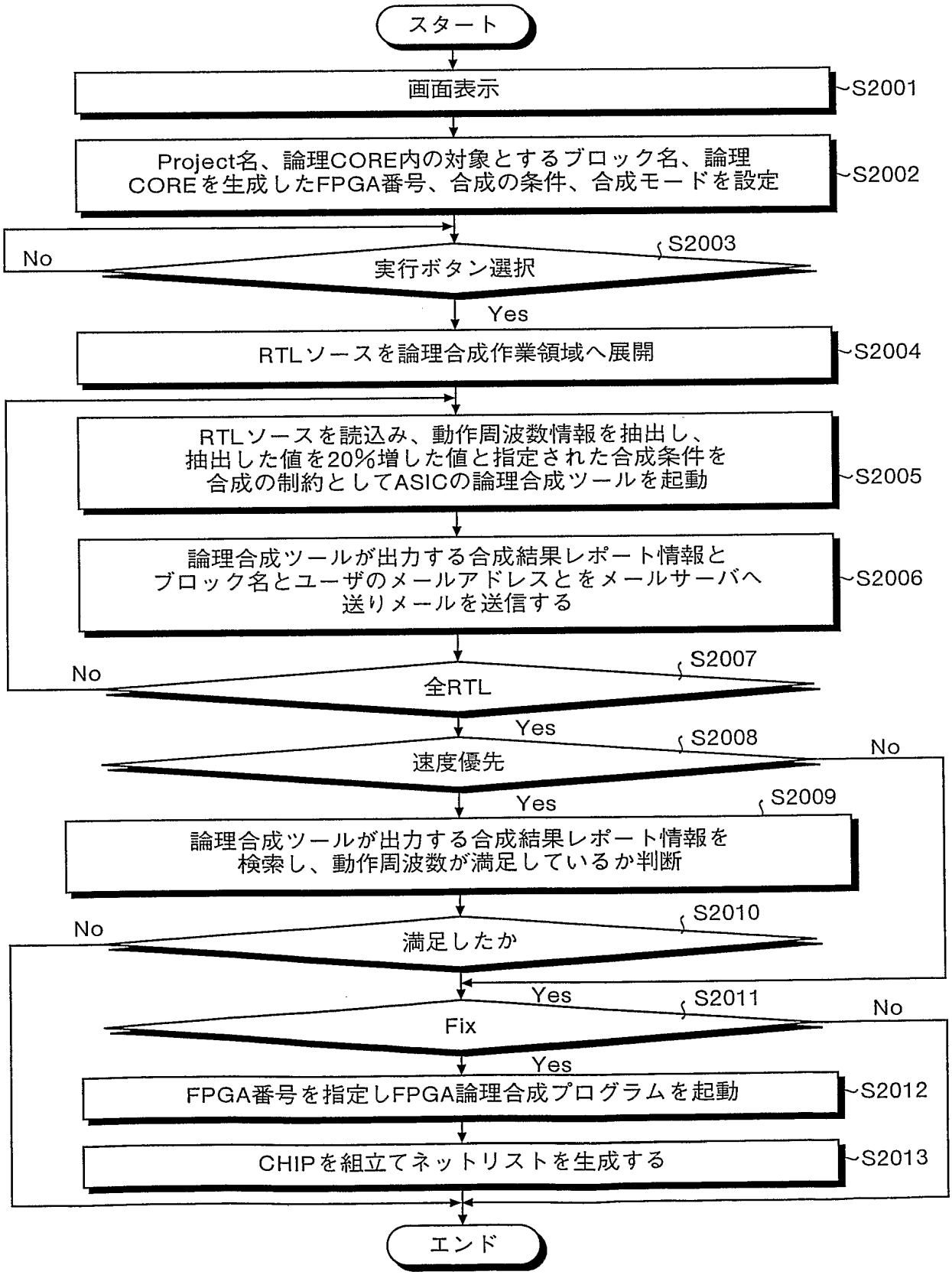
第29図


A
ファイルA読込み
S1710
インスタンス名とポート名を
全てメモリに格納
S1711
S1712
ポートは入力か？
No
Yes
S1713
出力インスタンス名と
その出力ポート名を抽出
S1715
当該ポートのネット名を
'インスタンス名+ポート名'
として定義し、当該ポートと
ネット名を対応させた接続
ネット情報を生成し、
ファイルBに出力
抽出した出力インスタンス名が
メモリにあるか検索
S1714
S1716
一致？
No
Yes
S1717
パッケージ端子
情報として
メモリに格納
当該ポートのネット名を出力元
'インスタンス名+出力元ポート名'と
して定義し、当該ポートとネット名を
対応させた接続ネット情報を生成し、
ファイルBに出力
S1718
S1719
全テーブルの
全ポート終了？
No
Yes
ASICの論理COREとして生成した時に出力したログファイルの
オープンポート情報に無いオープンポートを
パッケージ端子情報としてメモリに格納
S1720
論理COREのポート名、レンジ、入出力方向のパッケージ端子情報をメモリ
から読み出し論理COREのエンティティをファイルに出力する
S1721
論理COREのエンティティファイルとファイルBのネット情報を
読込み、結合する事で論理COREのネットリストを生成しファイル出力
S1722
エンド

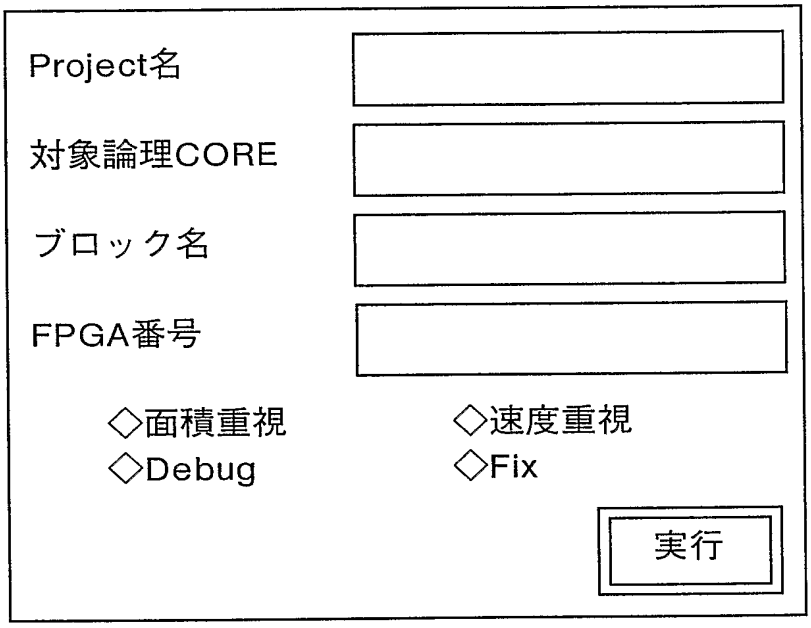
第30図

	IO数	ゲート規模	互換パッケージ
パッケージ名	XXX	GGGGG	ABCDEFG,YYYYYYY,...
.....	....	.....	..... ,...

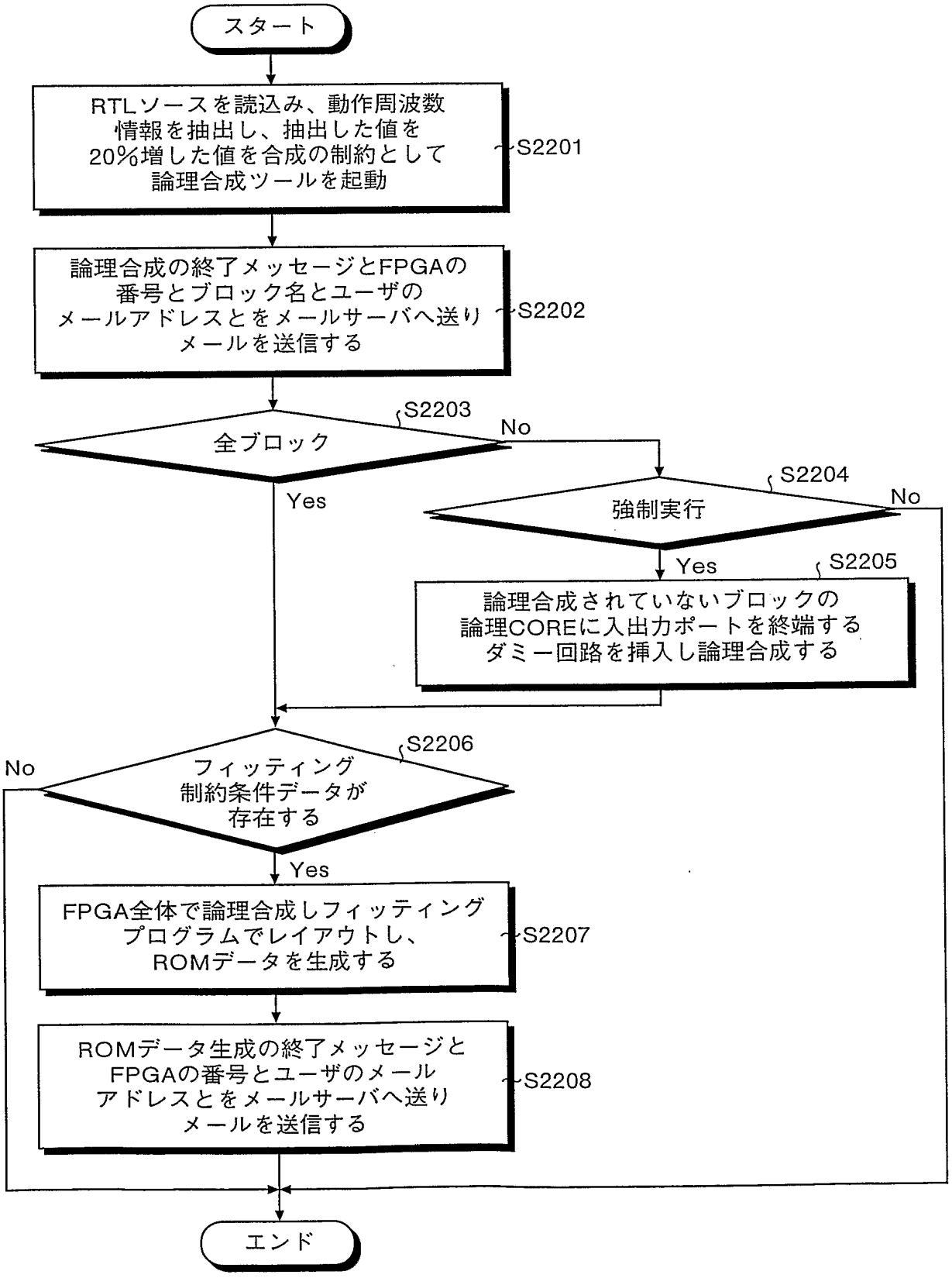
第31図


スタート
画面表示
S2001
Project名、論理CORE内の対象とするブロック名、論理COREを生成したFPGA番号、合成の条件、合成モードを設定
S2002
No
実行ボタン選択
S2003
Yes
RTLソースを論理合成作業領域へ展開
S2004
RTLソースを読込み、動作周波数情報を抽出し、抽出した値を20%増した値と指定された合成条件を合成の制約としてASICの論理合成ツールを起動
S2005
論理合成ツールが出力する合成結果レポート情報とブロック名とユーザのメールアドレスとをメールサーバへ送りメールを送信する
S2006
No
全RTL
S2007
Yes
速度優先
S2008
No
Yes
論理合成ツールが出力する合成結果レポート情報を検索し、動作周波数が満足しているか判断
S2009
No
満足したか
S2010
Yes
Fix
S2011
No
Yes
FPGA番号を指定しFPGA論理合成プログラムを起動
S2012
CHIPを組立てネットリストを生成する
S2013
エンド

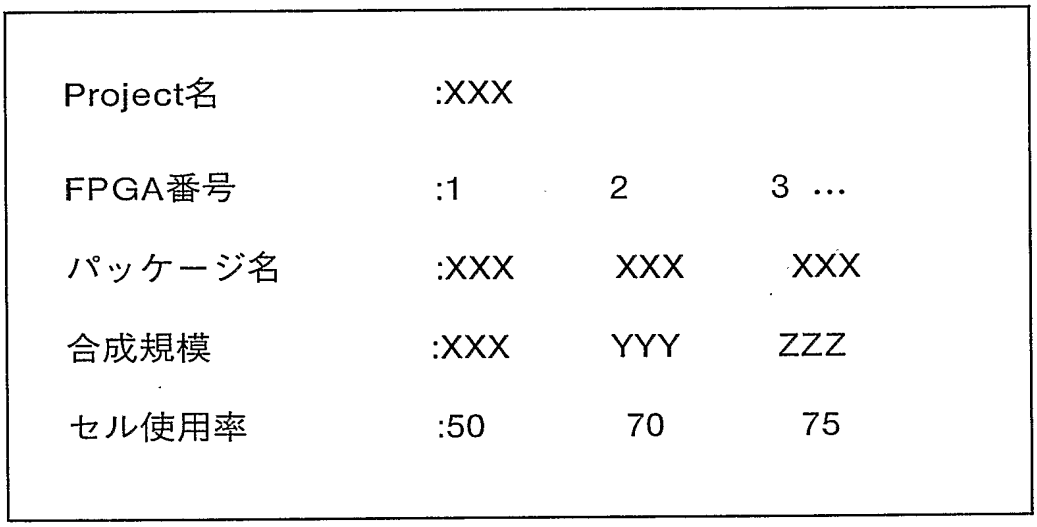
第32図



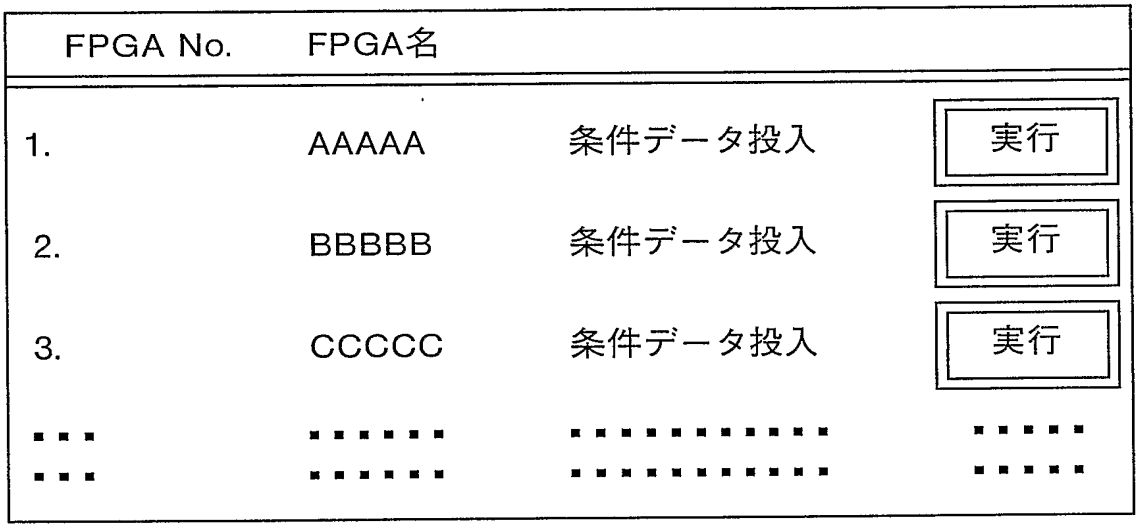
第33図


スタート
RTLソースを読込み、動作周波数情報を抽出し、抽出した値を20%増した値を合成の制約として論理合成ツールを起動
S2201
論理合成の終了メッセージとFPGAの番号とブロック名とユーザのメールアドレスとをメールサーバへ送りメールを送信する
S2202
全ブロック
S2203
No
Yes
強制実行
S2204
No
Yes
論理合成されていないブロックの論理COREに入出力ポートを終端するダミー回路を挿入し論理合成する
S2205
フィッティング制約条件データが存在する
S2206
No
Yes
FPGA全体で論理合成しフィッティングプログラムでレイアウトし、ROMデータを生成する
S2207
ROMデータ生成の終了メッセージとFPGAの番号とユーザのメールアドレスとをメールサーバへ送りメールを送信する
S2208
エンド

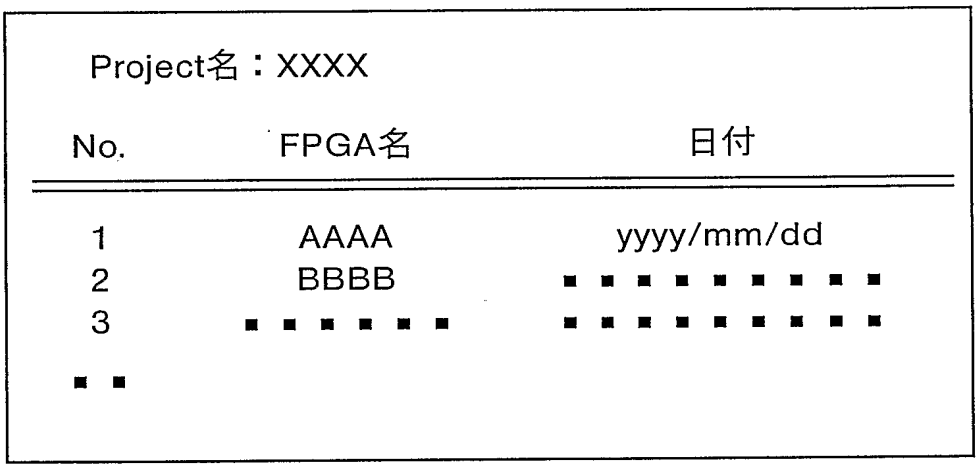
第34図


Project名 :XXX
FPGA番号 :1 2 3 …
パッケージ名 :XXX XXX XXX
合成規模 :XXX YYY ZZZ
セル使用率 :50 70 75

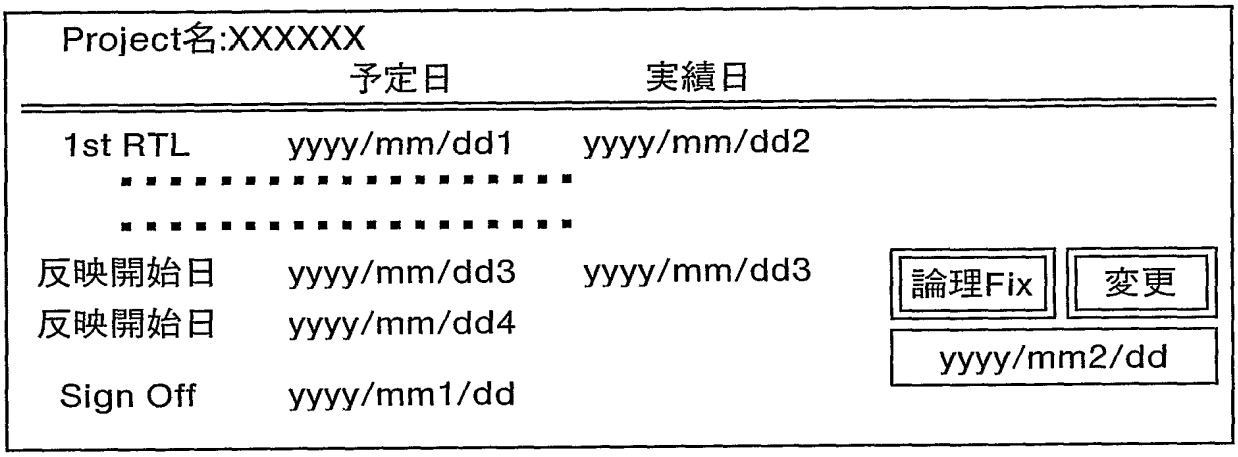
第35図


FPGA No.
FPGA名
1.
AAAAA
条件データ投入
実行
2.
BBBBB
条件データ投入
実行
3.
CCCCC
条件データ投入
実行

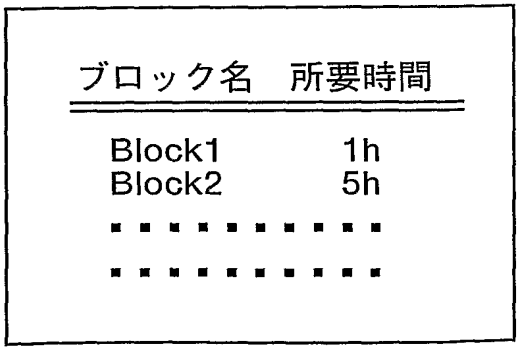
第36図



第37図


Project名:XXXXXX
予定日
実績日
1st RTL
yyyy/mm/dd1
yyyy/mm/dd2
反映開始日
yyyy/mm/dd3
yyyy/mm/dd3
論理Fix
変更
反映開始日
yyyy/mm/dd4
yyyy/mm2/dd
Sign Off
yyyy/mm1/dd

第38図


ブロック名 所要時間
Block1 1h
Block2 5h

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/04787

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F17/50, H03K19/173, H01L21/82

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F17/50, H03K19/173, H01L21/82

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
JSTplus FILE(JOIS)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP 2000-90142 A (Fujitsu Ltd.), 31 March, 2000 (31.03.00), Full text (Family: none)	1-16
Y	Tomoaki KOGA et al., "Mega Gate ASIC no Implement Sekkei", The Institute of Electronics, Information and Communication Engineers Gijutsu Kenkyu Hokoku, The Institute of Electronics, Information and Communication Engineers, 22 September, 1998 (22.09.98), Vol.98, No.287, pages 71 to 77 (VLD98-54)	1-16
Y	JP 8-194725 A (Fujitsu Ltd.), 30 July, 1996 (30.07.96), Full text (Family: none)	1-16

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
01 July, 2003 (01.07.03)	15 July, 2003 (15.07.03)
Name and mailing address of the ISA/ Japanese Patent Office	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/04787

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	Koichi HANDA et al., "FPGA/PLD-kan Kessen Check Shuho", The Institute of Electronics, Information and Communication Engineers Gijutsu Kenkyu Hokoku, The Institute of Electronics, Information and Communication Engineers, 28 November, 2002 (28.11.02), Vol.102, No.479, pages 115 to 120 (VLD2002-101)	1-16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/04787

Box I Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Performing logical synthesis in an integrated circuit design is a conventional technique.

Moreover, provision of an I/O buffer in an integrated circuit and using elements (including the I/O buffer) corresponding to the device technology belong to technical common sense.

Accordingly, the technical feature common to the inventions of claims 1-10, the inventions of claims 11-15, and the invention of claim 16 makes no contribution over the prior art and cannot be a special technical feature within the meaning of PCT Rule 13.2, second sentence.

(. continued to extra sheet)

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☒ No protest accompanied the payment of additional search fees.

Continuation of Box No.II of continuation of first sheet(1)

Consequently, the inventions of claims 1-10, the inventions of claims 11-15, and the invention of claim 16 do not satisfy the requirement of unity of invention.

国際調査報告	国際出願番号　ＰＣＴ／ＪＰ０３／０４７８７

A.　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Ｉｎｔ．Ｃｌ．[7]　Ｇ０６Ｆ１７／５０，Ｈ０３Ｋ１９／１７３，Ｈ０１Ｌ２１／８２

B.　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Ｉｎｔ．Ｃｌ．[7]　Ｇ０６Ｆ１７／５０，Ｈ０３Ｋ１９／１７３，Ｈ０１Ｌ２１／８２

最小限資料以外の資料で調査を行った分野に含まれるもの

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

ＪＳＴｐｌｕｓファイル（ＪＯＩＳ）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y	JP 2000-90142 A（富士通株式会社）2000.03.31, 全文，（ファミリーなし）	1-16
Y	古賀智昭，外６名，‘メガゲートＡＳＩＣのインプリメント設計’， 電子情報通信学会技術研究報告，電子情報通信学会，1998.09.22, Vol.98, No.287, p.71-77 (VLD98-54)	1-16
Y	JP 8-194725 A（富士通株式会社）1996.07.30, 全文，（ファミリーなし）	1-16

☒ Ｃ欄の続きにも文献が列挙されている。　　☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
０１．０７．０３	15.07.03

国際調査機関の名称及びあて先	特許庁審査官（権限のある職員）		
日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	早川　学	５Ｈ	９６５２
	電話番号　０３－３５８１－１１０１　内線　３５３１		

様式ＰＣＴ／ＩＳＡ／２１０（第２ページ）（１９９８年７月）

Ｃ（続き）． 関連すると認められる文献

引用文献のカテゴリー＊	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
PX	反田浩一，外１名，‘ＦＰＧＡ／ＰＬＤ間結線チェック手法’，電子情報通信学会技術研究報告，電子情報通信学会，2002.11.28，Vol.102，No.479，p.115-120（VLD2002-101）	1-16

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ０３／０４７８７

第Ⅰ欄　請求の範囲の一部の調査ができないときの意見（第１ページの２の続き）

法第８条第３項（ＰＣＴ１７条(2)(a)）の規定により、この国際調査報告は次の理由により請求の範囲の一部について作成しなかった。

1．☐ 請求の範囲 _____ は、この国際調査機関が調査をすることを要しない対象に係るものである。つまり、

2．☐ 請求の範囲 _____ は、有意義な国際調査をすることができる程度まで所定の要件を満たしていない国際出願の部分に係るものである。つまり、

3．☐ 請求の範囲 _____ は、従属請求の範囲であってＰＣＴ規則6.4(a)の第２文及び第３文の規定に従って記載されていない。

第Ⅱ欄　発明の単一性が欠如しているときの意見（第１ページの３の続き）

次に述べるようにこの国際出願に二以上の発明があるとこの国際調査機関は認めた。

集積回路設計において、論理合成を行うことは慣用技術である。
また、集積回路において、Ｉ／Ｏバッファを設けること、及び、素子（Ｉ／Ｏバッファを含む。）をデバイス・テクノロジに対応したものとすることは技術常識である。
してみれば、請求の範囲１－１０に係る発明と、請求の範囲１１－１５に係る発明と、請求の範囲１６に係る発明との間に共通する事項は、先行技術の域を出ないから、ＰＣＴ規則１３．２の第２文の意味において、特別な技術的特徴ではない。
したがって、請求の範囲１－１０に係る発明と、請求の範囲１１－１５に係る発明と、請求の範囲１６に係る発明とは、単一性の要件を満たしていない。

1．☒ 出願人が必要な追加調査手数料をすべて期間内に納付したので、この国際調査報告は、すべての調査可能な請求の範囲について作成した。

2．☐ 追加調査手数料を要求するまでもなく、すべての調査可能な請求の範囲について調査することができたので、追加調査手数料の納付を求めなかった。

3．☐ 出願人が必要な追加調査手数料を一部のみしか期間内に納付しなかったので、この国際調査報告は、手数料の納付のあった次の請求の範囲のみについて作成した。

4．☐ 出願人が必要な追加調査手数料を期間内に納付しなかったので、この国際調査報告は、請求の範囲の最初に記載されている発明に係る次の請求の範囲について作成した。

追加調査手数料の異議の申立てに関する注意

☐ 追加調査手数料の納付と共に出願人から異議申立てがあった。

☒ 追加調査手数料の納付と共に出願人から異議申立てがなかった。